E66

Shared Queues Implementation Considerations

Bill Stillwell
stillwel@us.ibm.com

St. Louis, MO **Sept. 30 - Oct. 3, 2002**

RETURN TO INDEX

Abstract

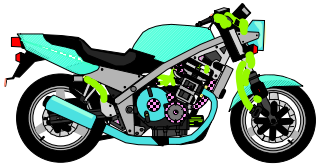
IMS Shared Queues support was introduced into the product in Version 6. Since then, each IMS release has added some functionality and usability to the feature.

This presentation includes a review of Shared Queues implementation and benefits, and highlights the changes in Versions 7 & 8.

WHY use Shared Queues?

Availability

Capacity



Performance

Workload Balancing

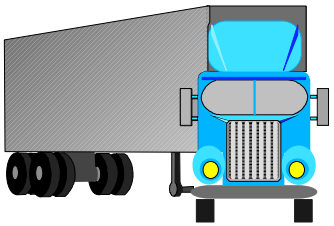
Some Terminology

Availability

- *Access, by the end user, to the data and facilities* of IMS necessary to perform that end user's business function

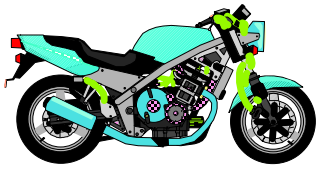
Capacity

- The ability of the IMS complex (one or more IMSs) to *process the total workload* within the required time



Performance

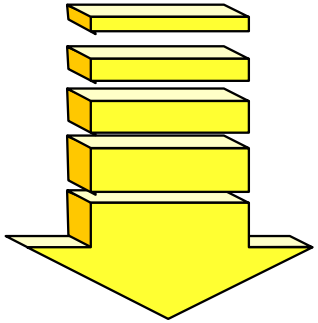
- The *response* that an individual user gets when submitting a work request (transaction, batch job) to IMS



How Can We Get All Three?

Workload Balancing / Workload Distribution

- The *distribution of the transaction and/or batch workload* across multiple IMSs in an IMSplex to achieve ...



Availability

- If one IMS fails, others remain available to process workload

Capacity

- More resources to do the work; utilization of all available resources

Performance

- Users' work doesn't sit in queue due to lack of capacity

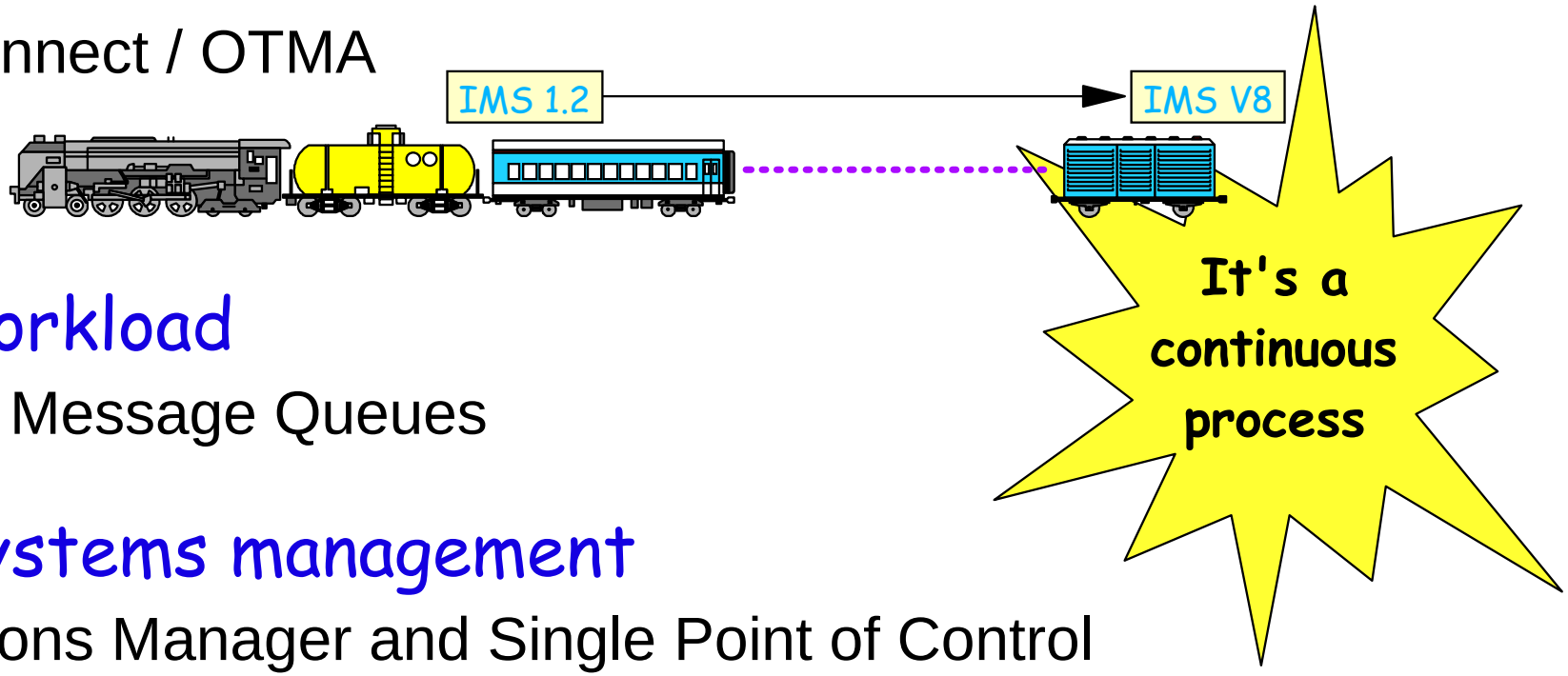
Enabling Workload Balancing in IMS

Shared data

- Full function and fast path databases

Shared network

- VTAM Generic Resources
- IMS Connect / OTMA



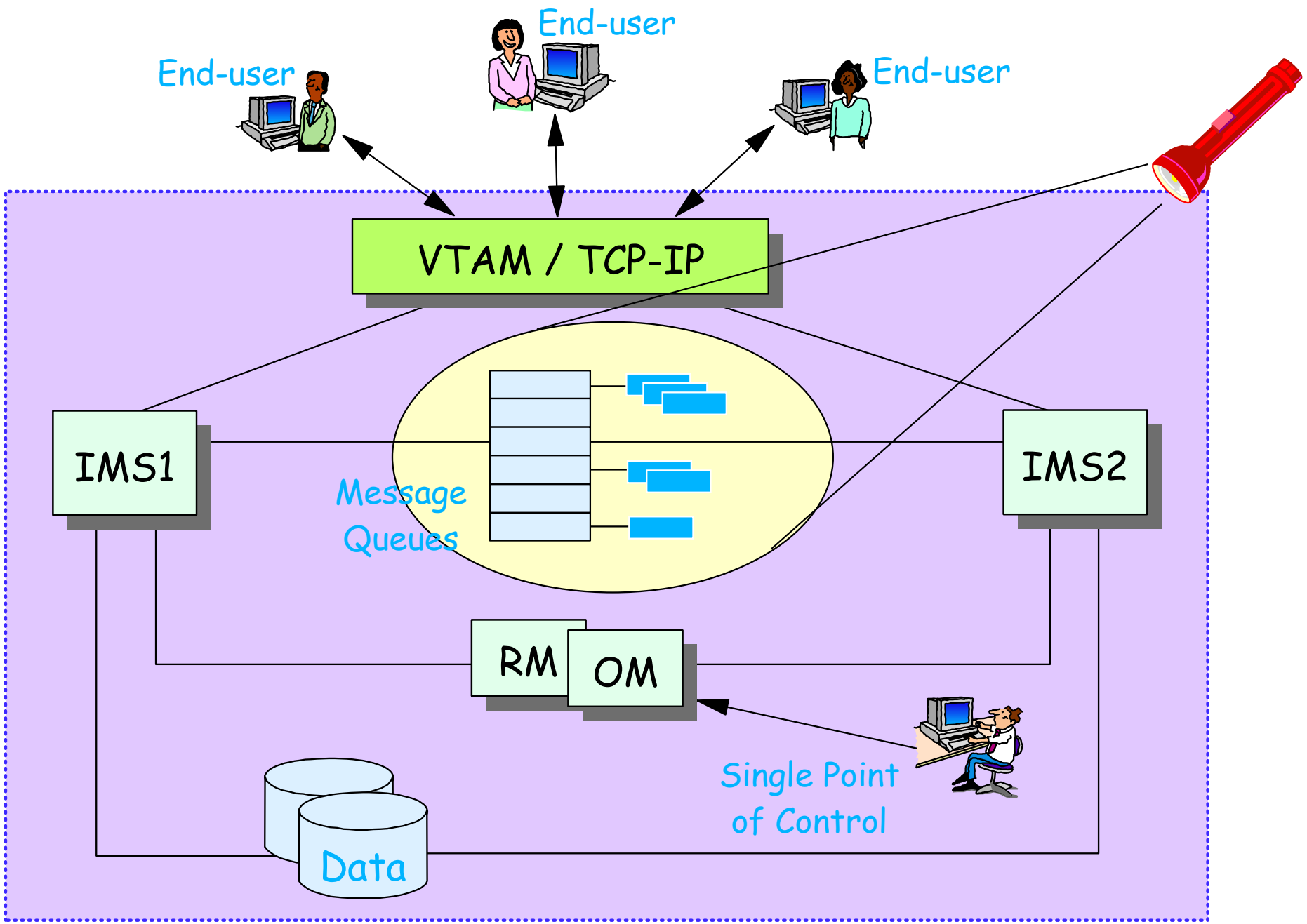
Shared workload

- Shared Message Queues

Shared systems management

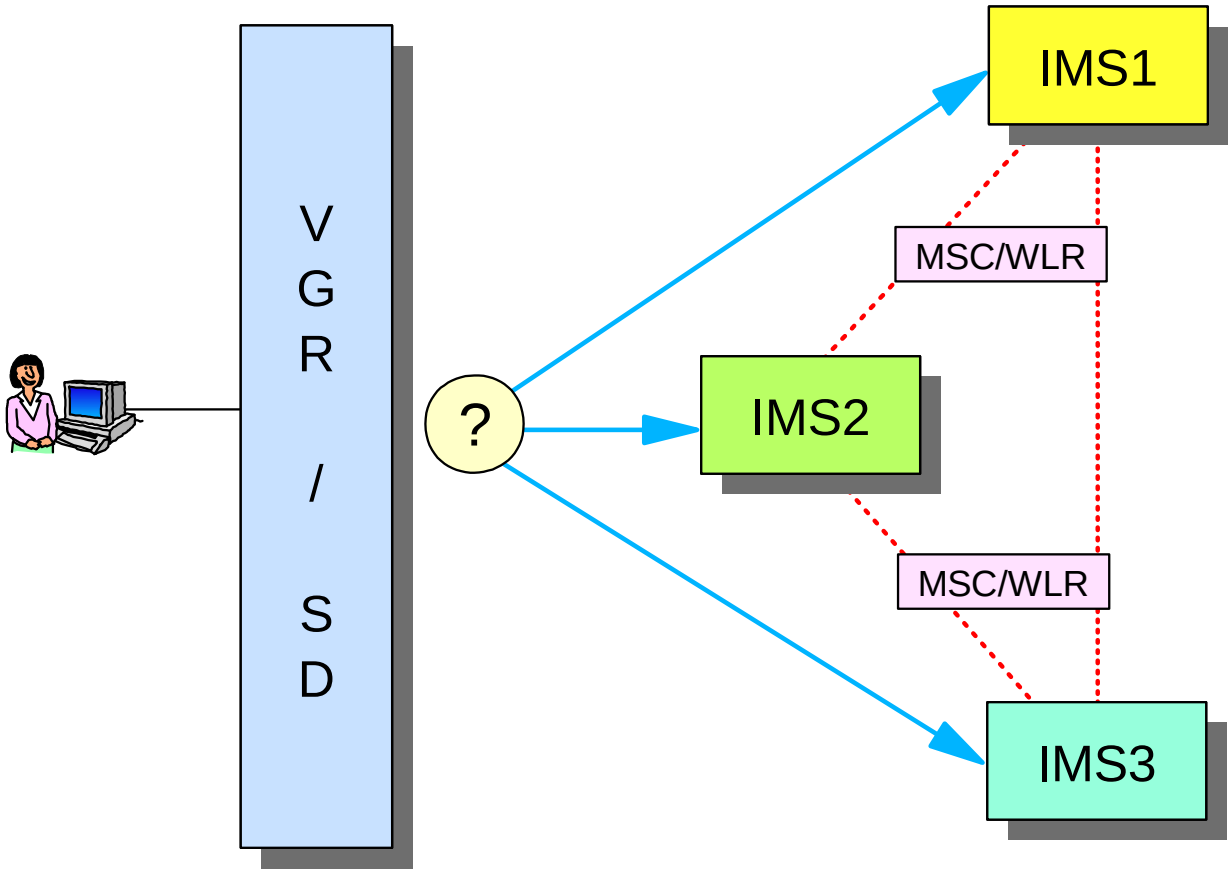
- Operations Manager and Single Point of Control
- Resource Manager, Sysplex Terminal Management, and Coordinated Global Online Change

Single End-user Image



Where To Do the Work?

When pushing down the workload, we can only hope that it is balanced

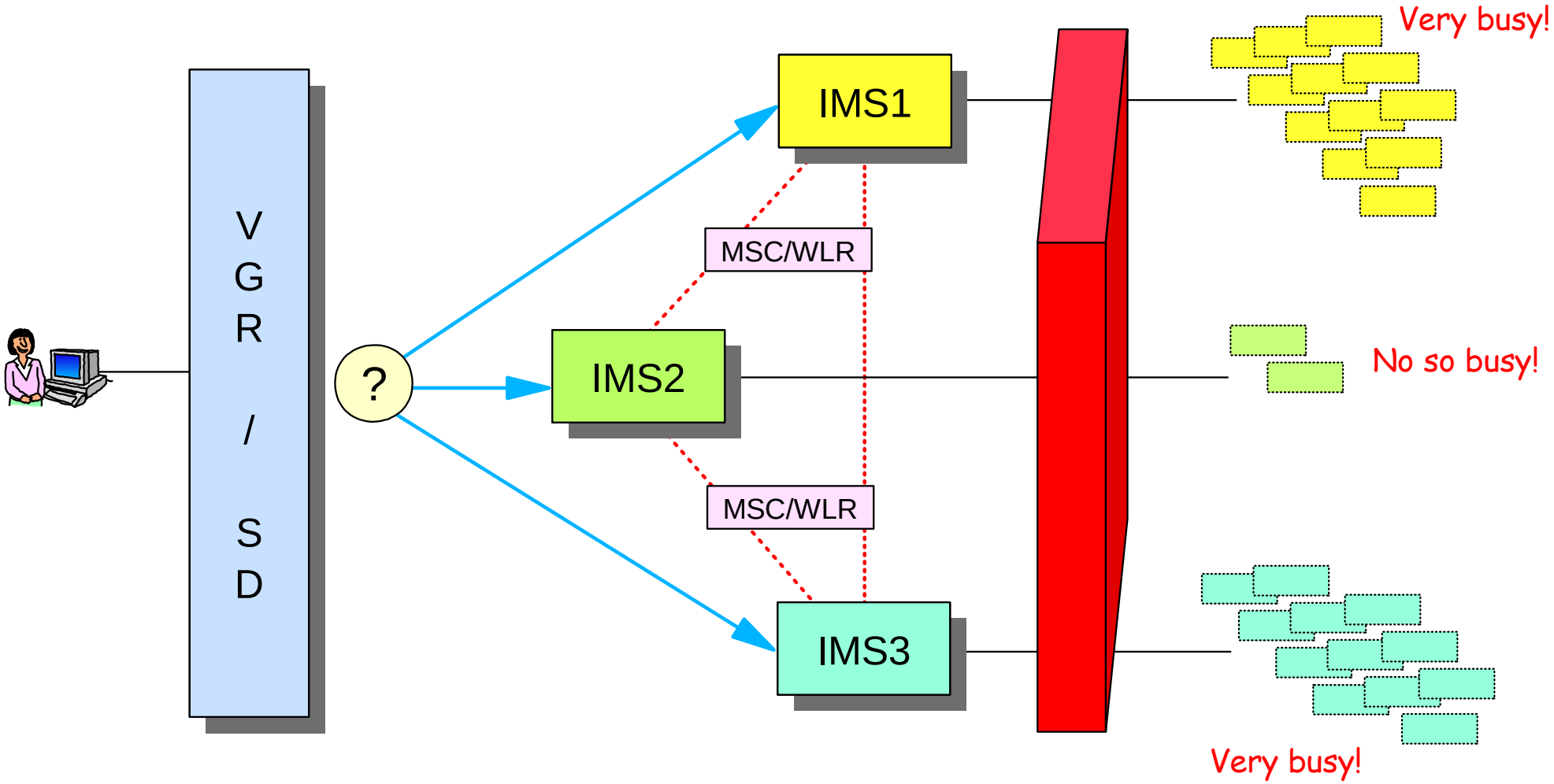


MSC and Workload Router, are examples of workload distribution tools that use the push-down technique.

IMS Connect, VTAM Generic resources, the Sysplex Distributor, or just telling end-users which IMS to log on to are other examples.

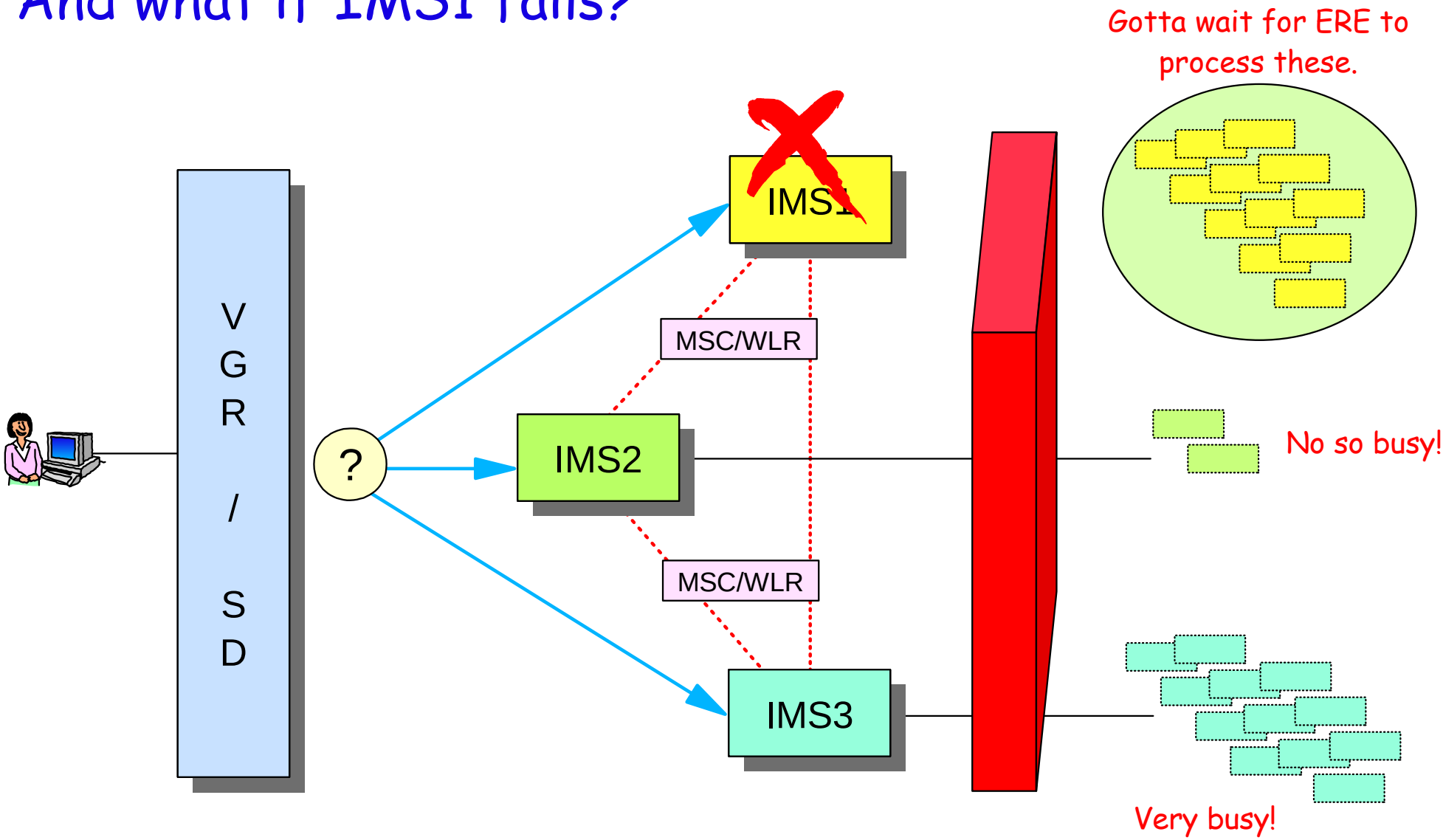
Where ...

But! We can't see the work already in the queue!



Workload Balancing . . .

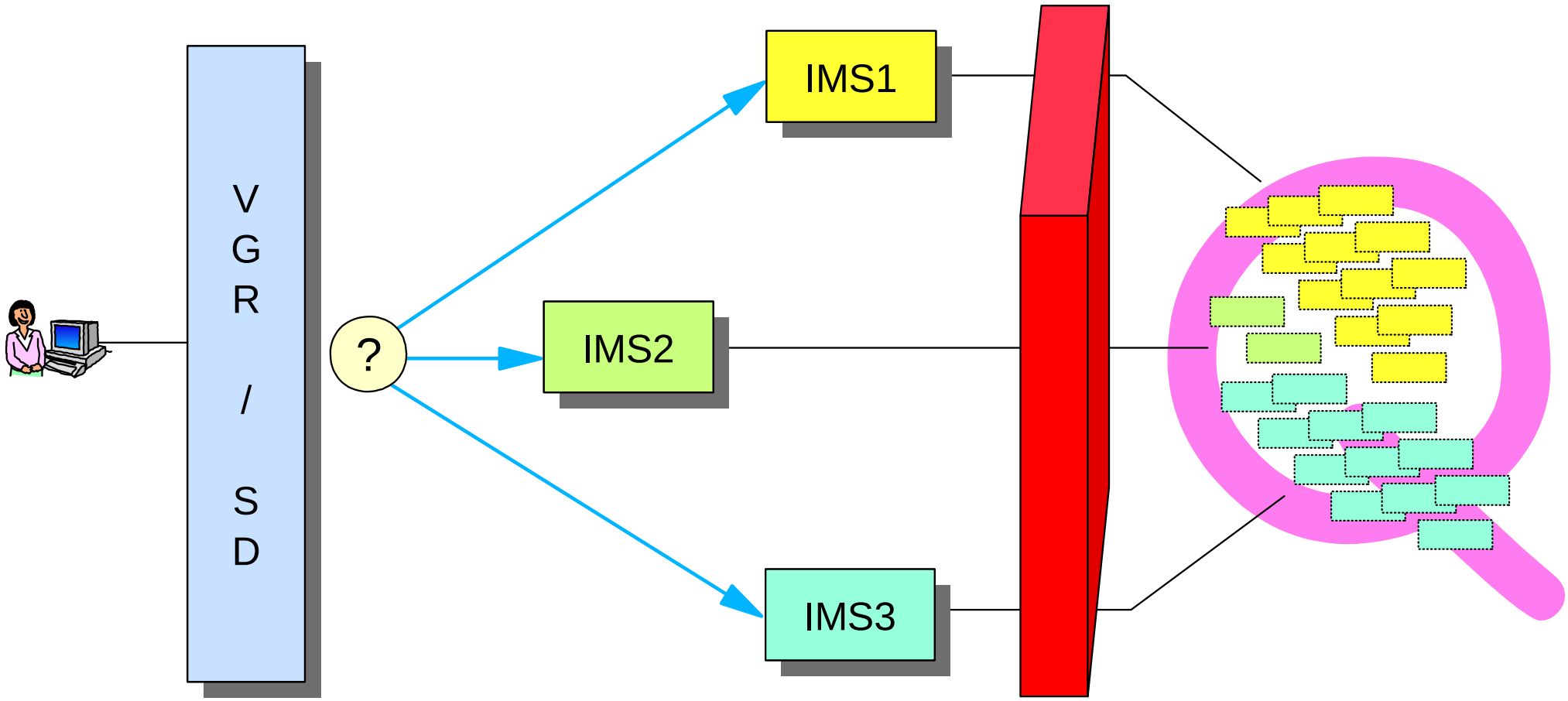
And what if IMS1 fails?



Workload Balancing

What we need is a single ***shared queue***

- Still can't see the workload, but it doesn't matter



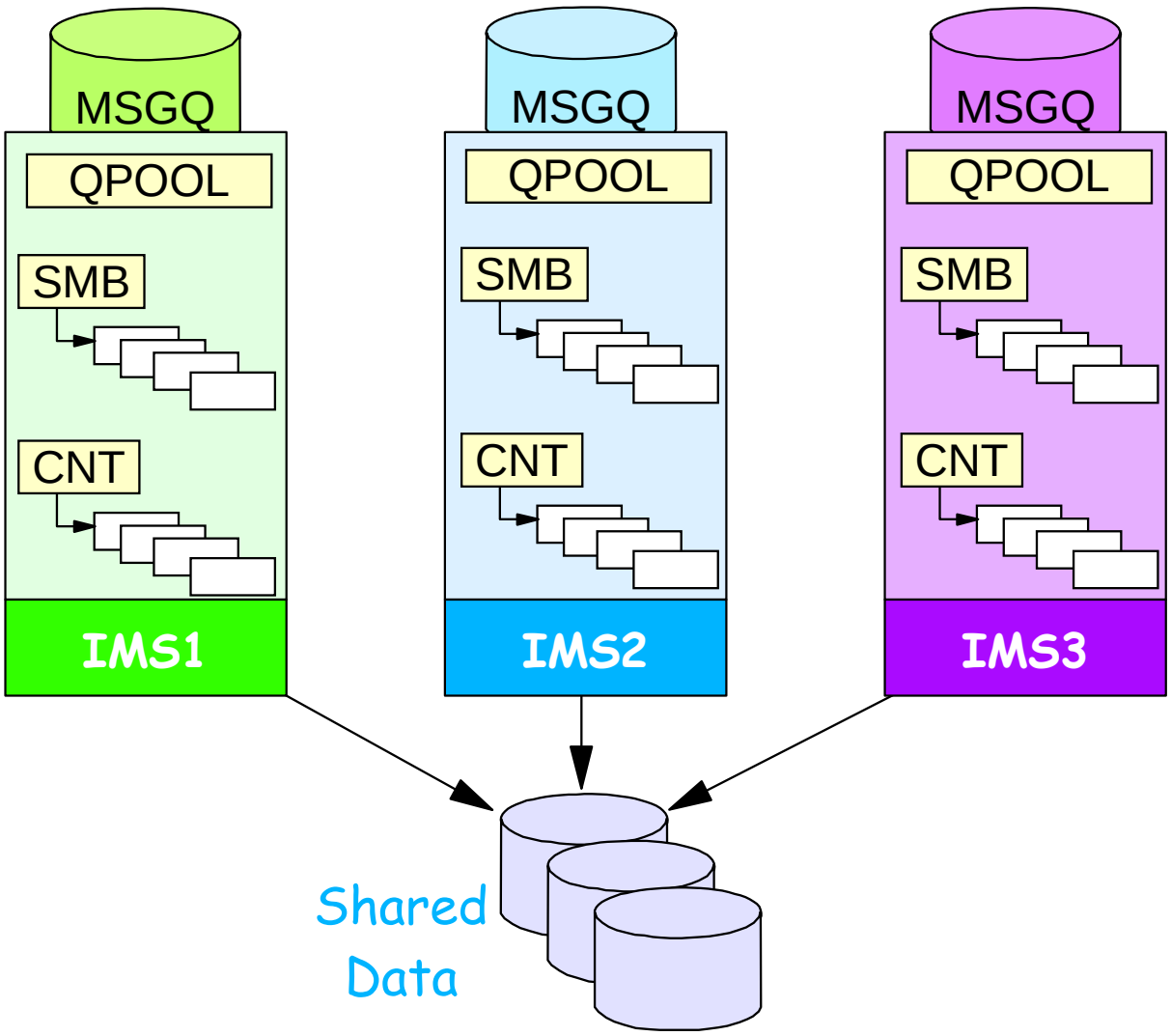
What Are IMS Shared Queues?

Shared queues are a set of input and output message queues which can be shared by multiple IMS systems

An extension of IMS's exploitation of parallel sysplex functionality

- Provides single (IMS) system image to end-users
  - End-user can enter transaction to any IMS in shared queues group
- Allows workload to be distributed across multiple IMS images
  - Input transaction can execute on any IMS in shared queues group
- Increases capacity of IMS application system
  - Up to 32 IMS systems on 32 MVS images can combine to process workload
- Improves availability of IMS application system
  - If any IMS goes down, workload can be assumed by surviving IMSs

Message Queuing without SQ



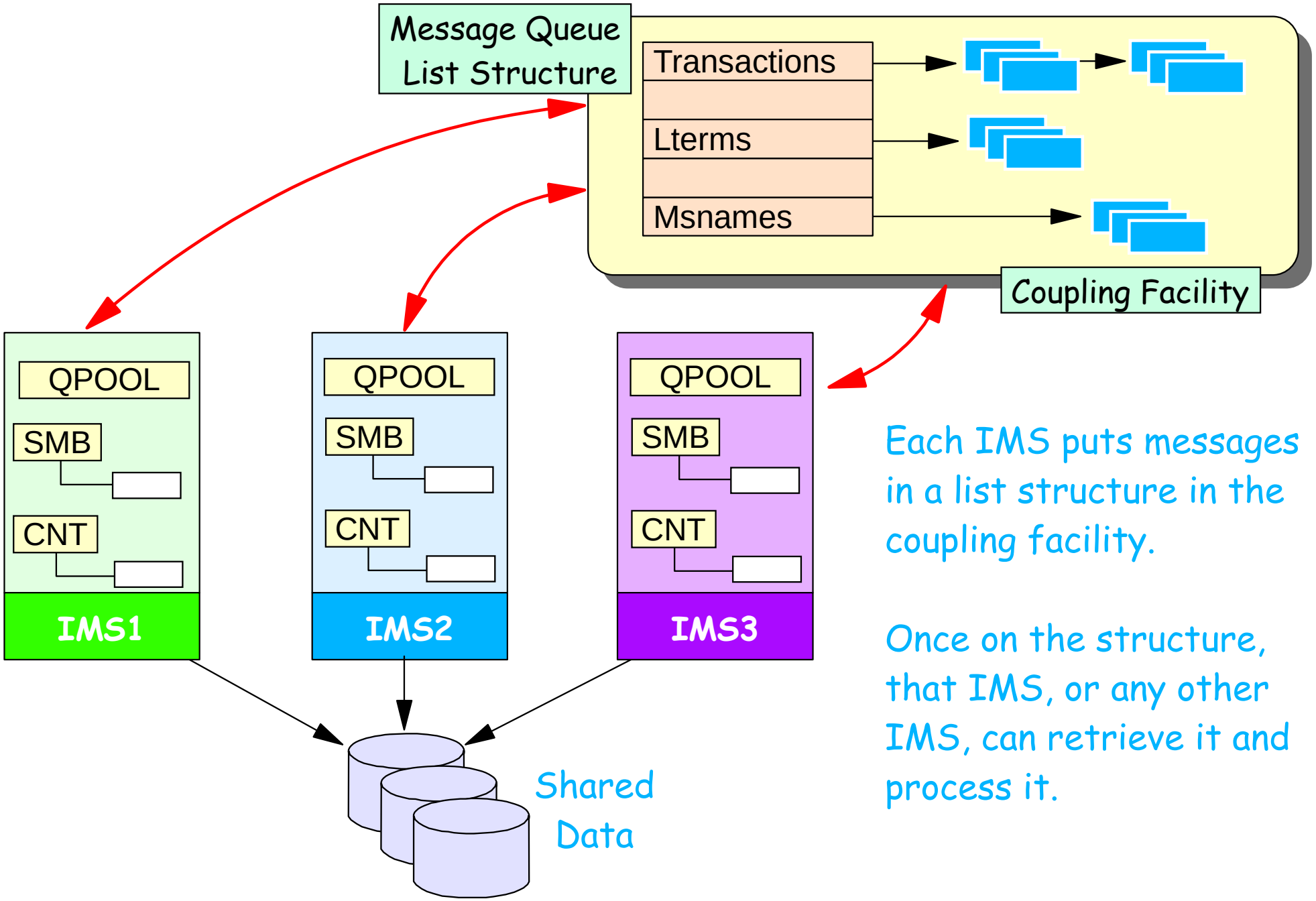
IMSs can shared data but each IMS has exclusive use of its own queues (workload).

Workload balancing between systems is a user responsibility.

If one IMS is overloaded others can't help. If one IMS fails, others can't assume workload.

User is dependent on a single IMS.

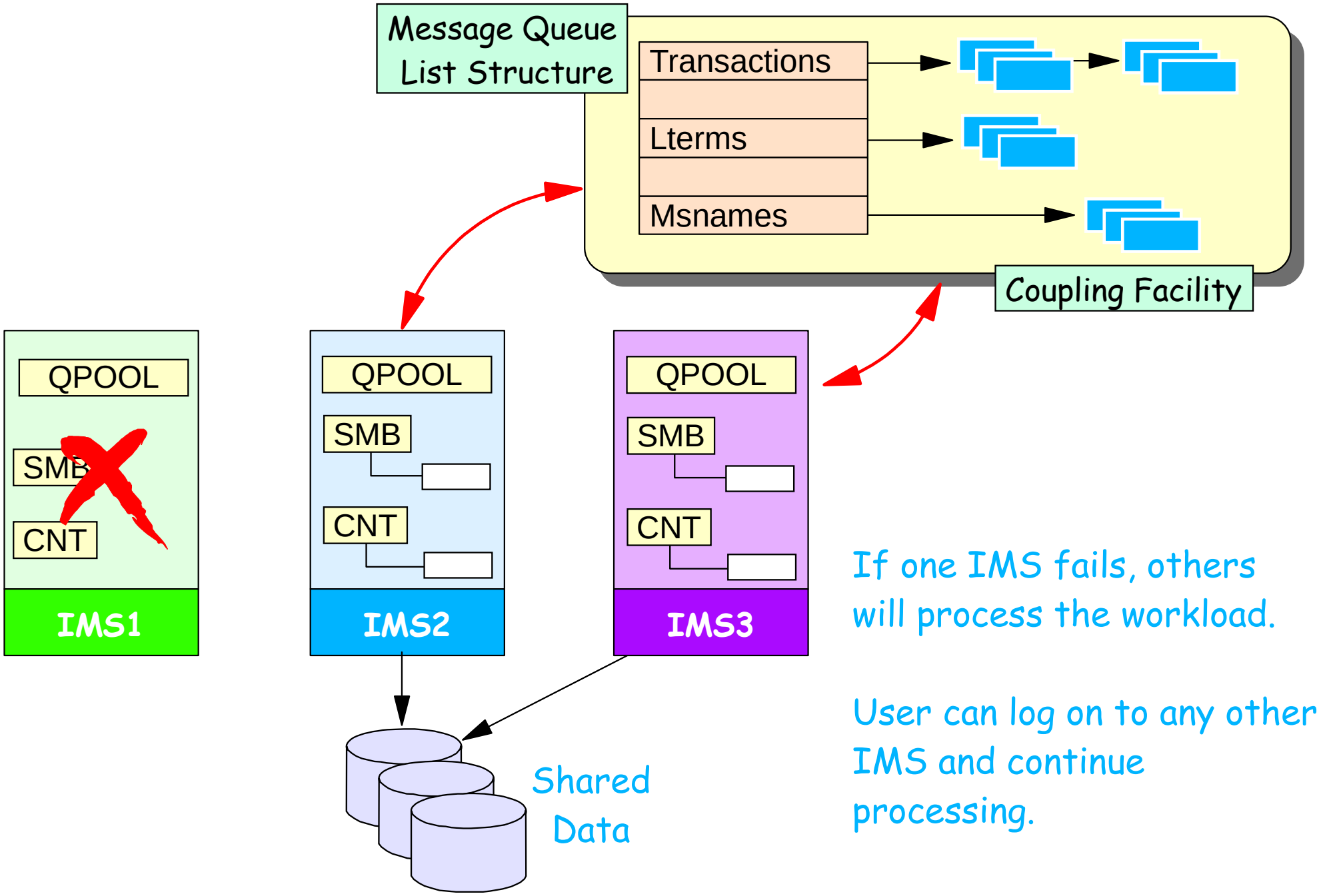
Message Queuing with SQ



Each IMS puts messages in a list structure in the coupling facility.

Once on the structure, that IMS, or any other IMS, can retrieve it and process it.

Message Queuing with SQ ...



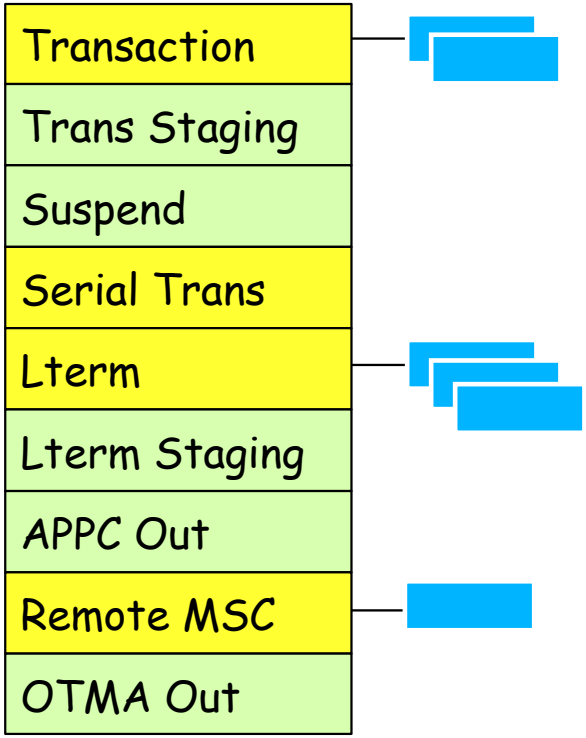
If one IMS fails, others will process the workload.

User can log on to any other IMS and continue processing.

IMS Shared Queues

Shared Message Queues (MSGQ)

- IMS full function messages
  available to multiple IMS subsystems
- Multiple input queue types
  - Transaction, Serial Transaction
  - Suspended Transaction
- Multiple output queue types
  - LTERM, APPC-Out, OTMA-Out, Remote MSC

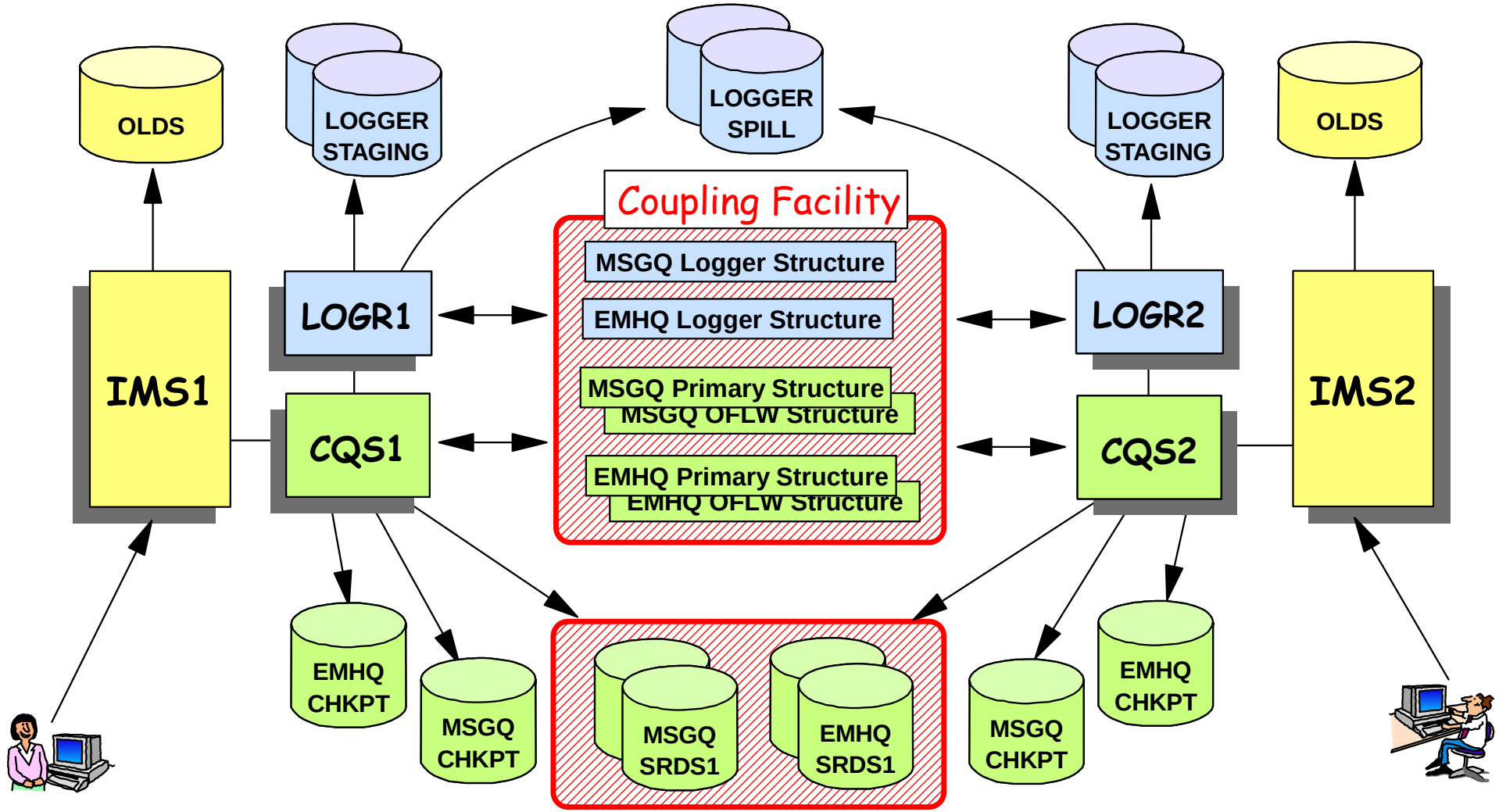


Shared EMH Queues (EMHQ)

- Fast path EMH messages
  available to multiple IMS subsystems
- Single input queue type
  - Program (PSB/BALG)
- Single output queue type
  - LTERM

Messages are queued according to type and name.

Shared Queues Components



Common CQS Data Sets

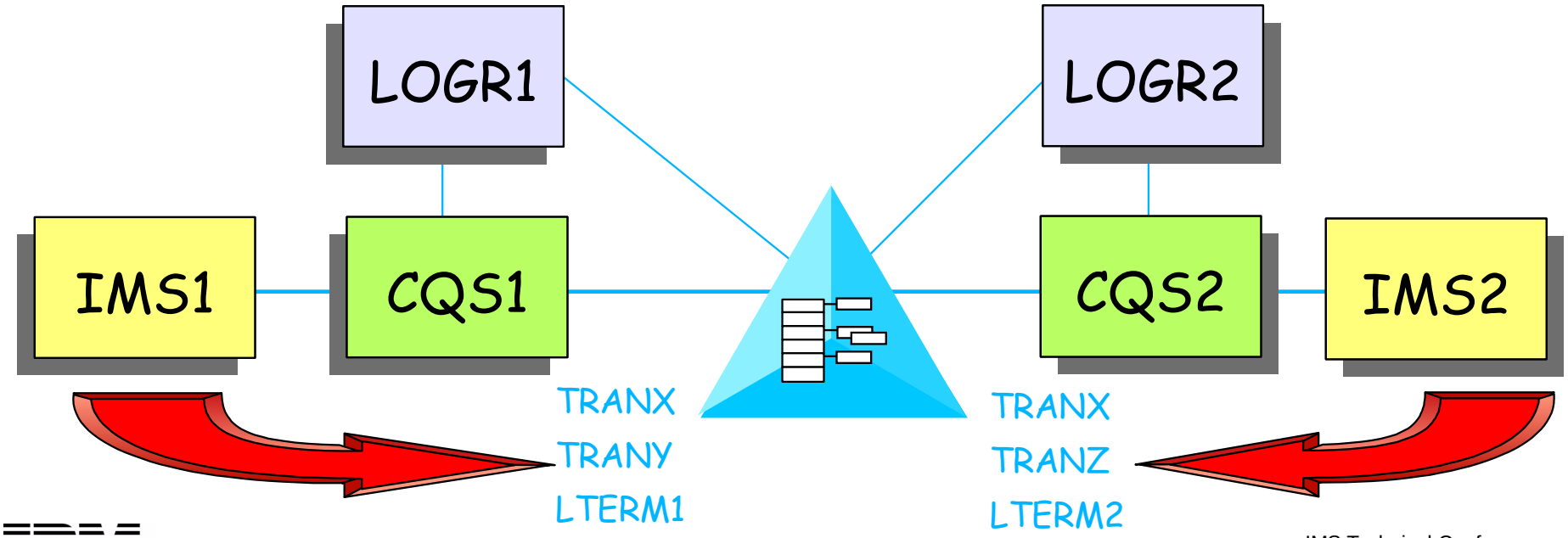
Implementation of Shared Queues

IMS

- *Connects to MSGQ and to EMHQ* list structures
  - Connection is through a Common Queue Server (CQS)
- *Registers interest* in specific queues
  - Indicates that IMS is *capable of processing* a message on that queue

CQS

- Services requests from IMS to queue or retrieve messages
  - Logs structure activity using System Logger



Implementation ...

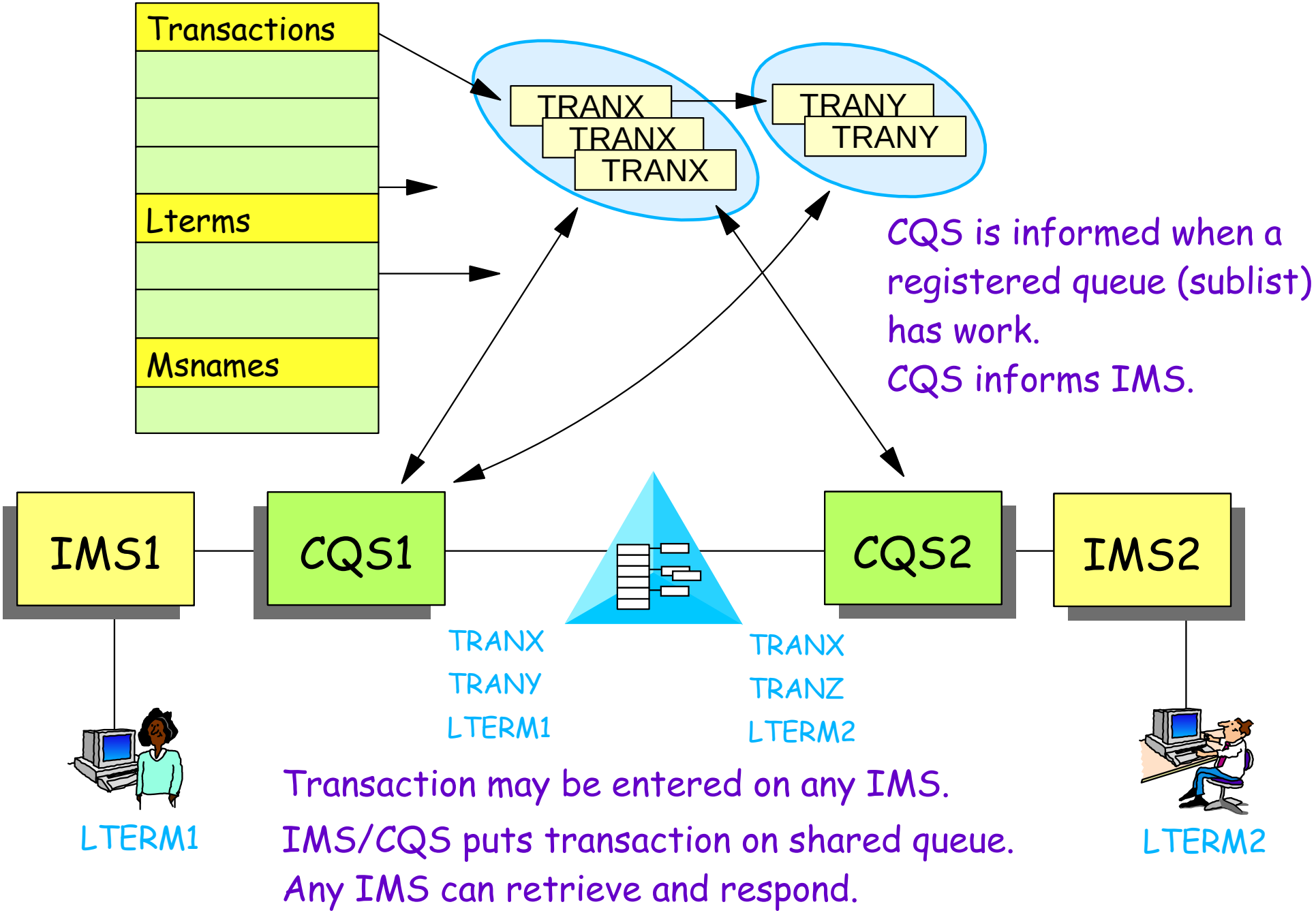
Registering interest

- IMS registers interest in messages it is able to process
  - Input transactions (can schedule - defined, not stopped)
  - Output messages (can deliver - in session with terminal)
  - Remote queue (can deliver - MSC link to remote IMS is active)

Message handling

- When IMSn receives an input message it may place it on a shared input ready queue (e.g., FF transaction or FP program)
  - *Any IMS with registered interest in that transaction* may retrieve it from the shared queue and process it
  - Should be multiple (all) IMSs with registered interest
- When IMSn has an output message, it places it on a shared output ready queue (e.g., Lterm, Remote MSC)
  - *Any IMS with registered interest in that Lterm or Link* may retrieve it from the queue and send it
  - Should be only one IMS with registered interest

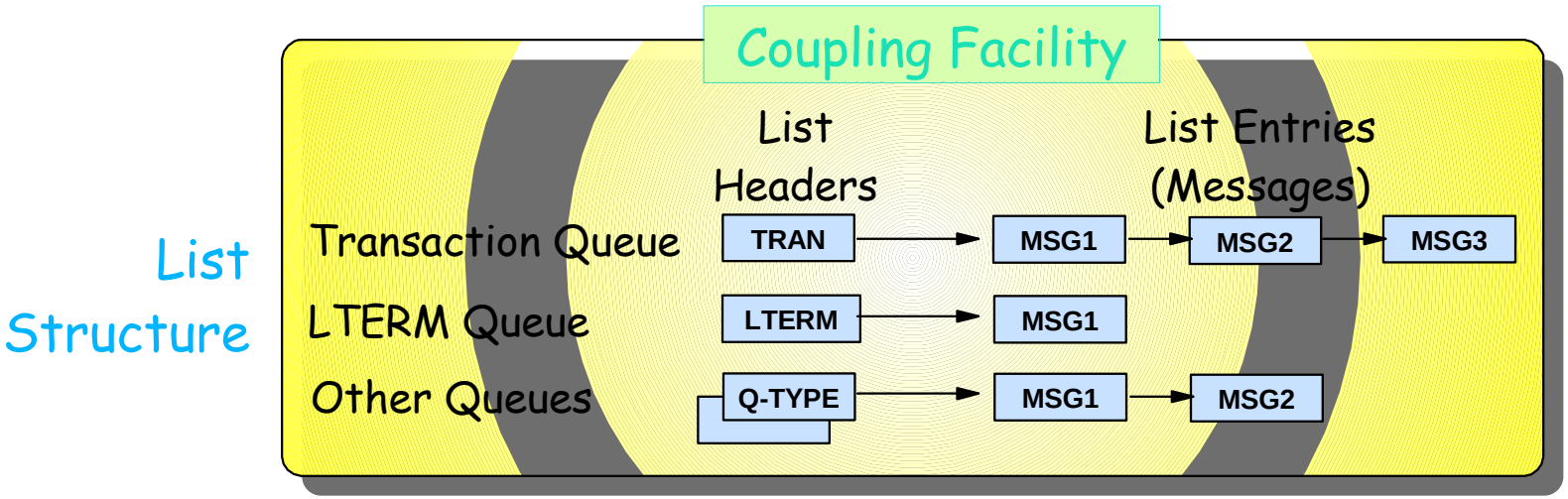
IMS Shared Queues



Shared Queues Structures

Queues are maintained in List Structures in the Coupling Facility

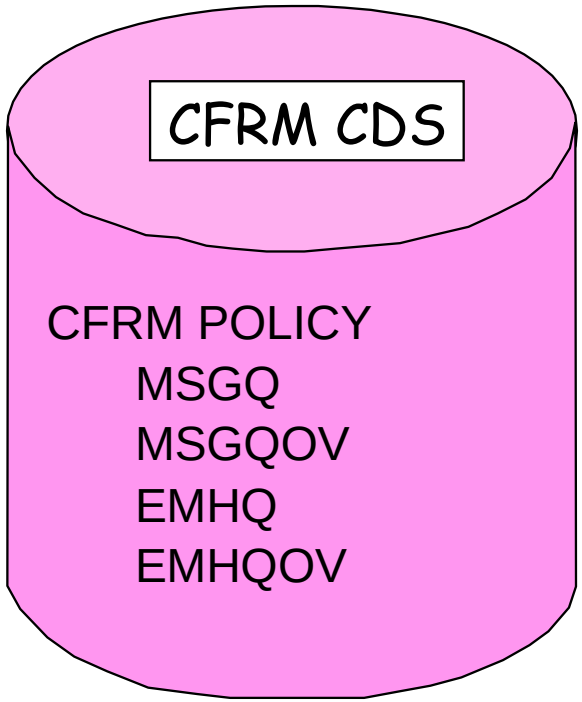
- Defined in CFRM Policy
- One primary structure for FF messages
  - Optional overflow structure
- One primary structure for EMH messages
  - Optional overflow structure

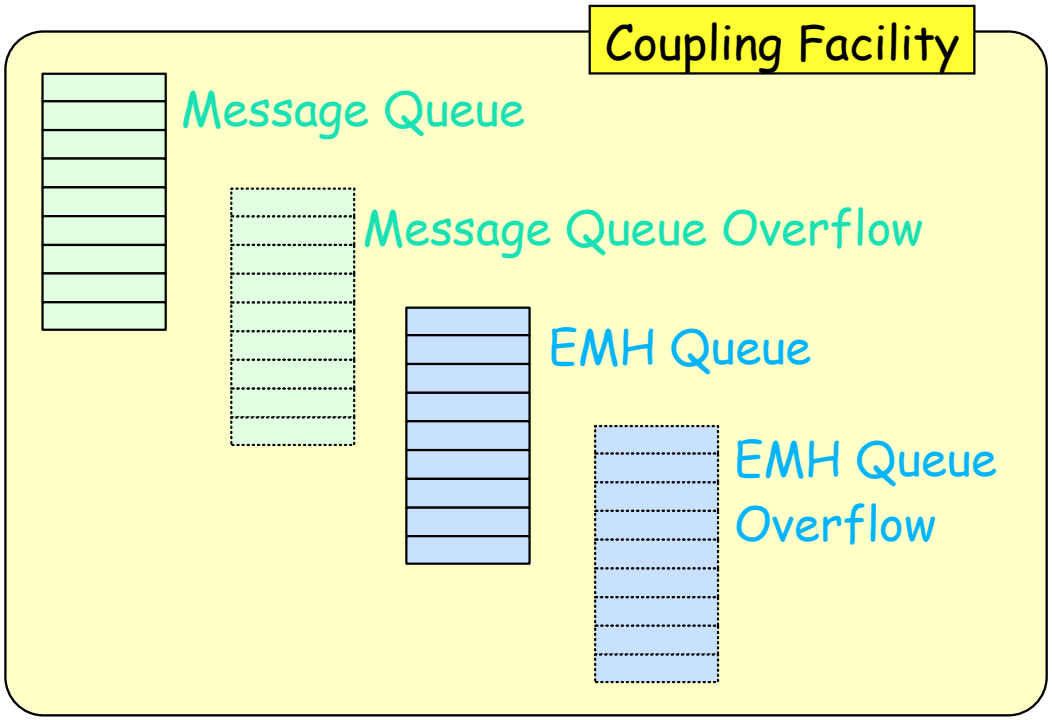


Shared Queues Structures

Queues are maintained in List Structures in the Coupling Facility

- Defined in CFRM Policy
- One primary structure for FF messages
  - Optional overflow structure
- One primary structure for EMH messages (if FP enabled)
  - Optional overflow structure

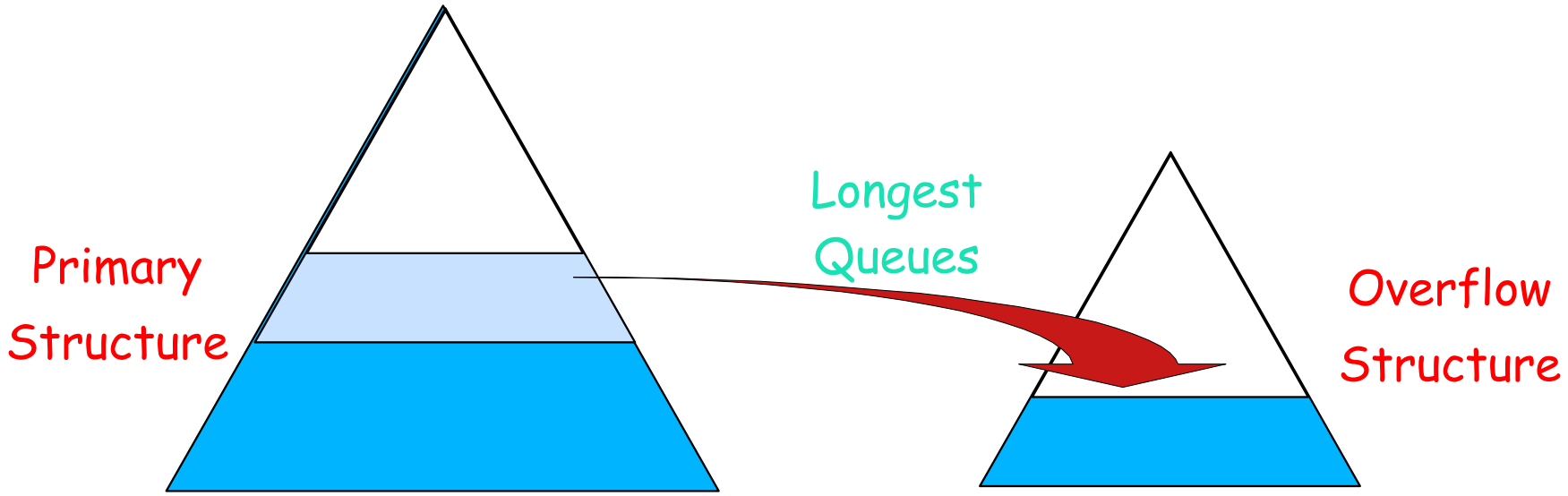




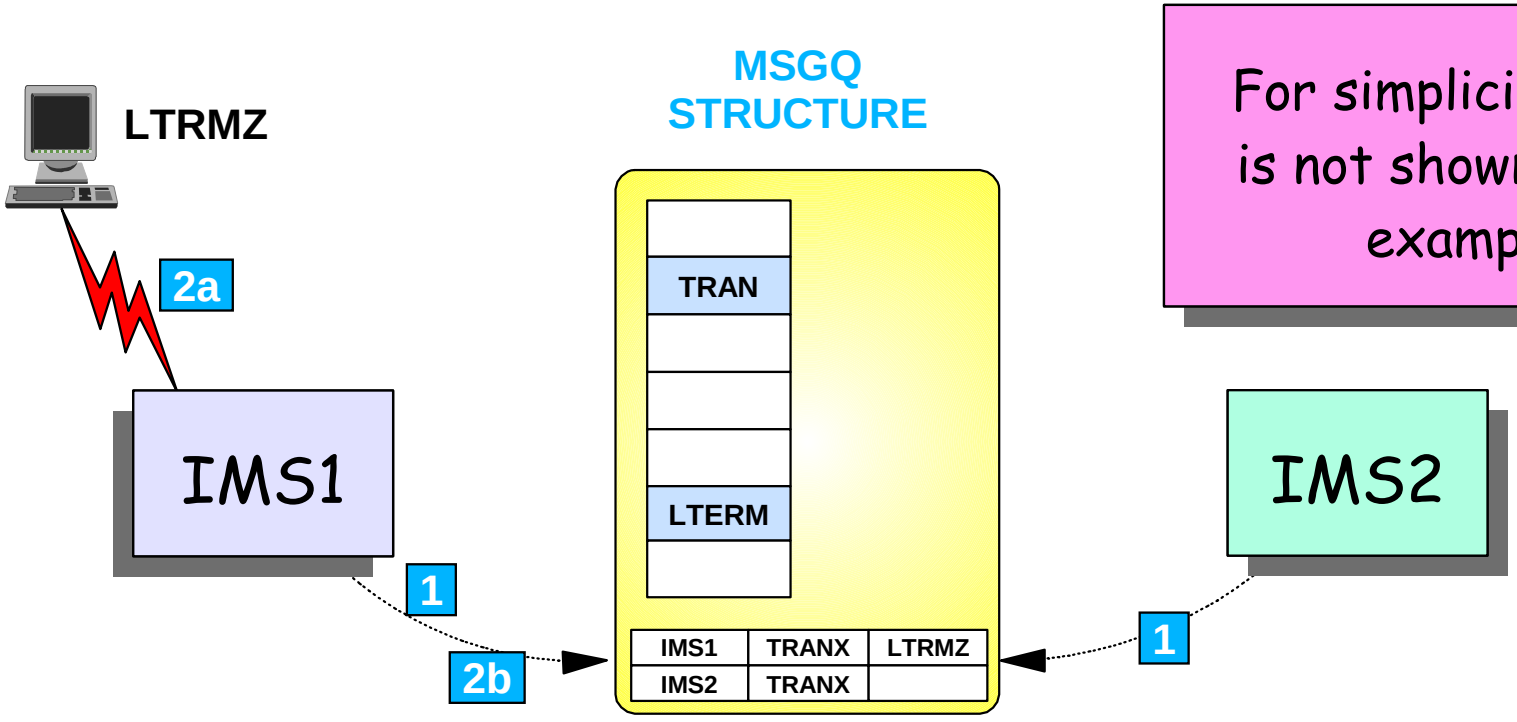
Overflow Processing

Overflow Structure

- Optional
  - Structure is defined in CFRM Policy
- Contains messages for QNAMEs (Transaction codes, Lterm names, ...) which have been selected for overflow
- Relieves storage constraint in Primary Structure
- *Eliminates cause of most U113 abends*



Shared Message Queue Example



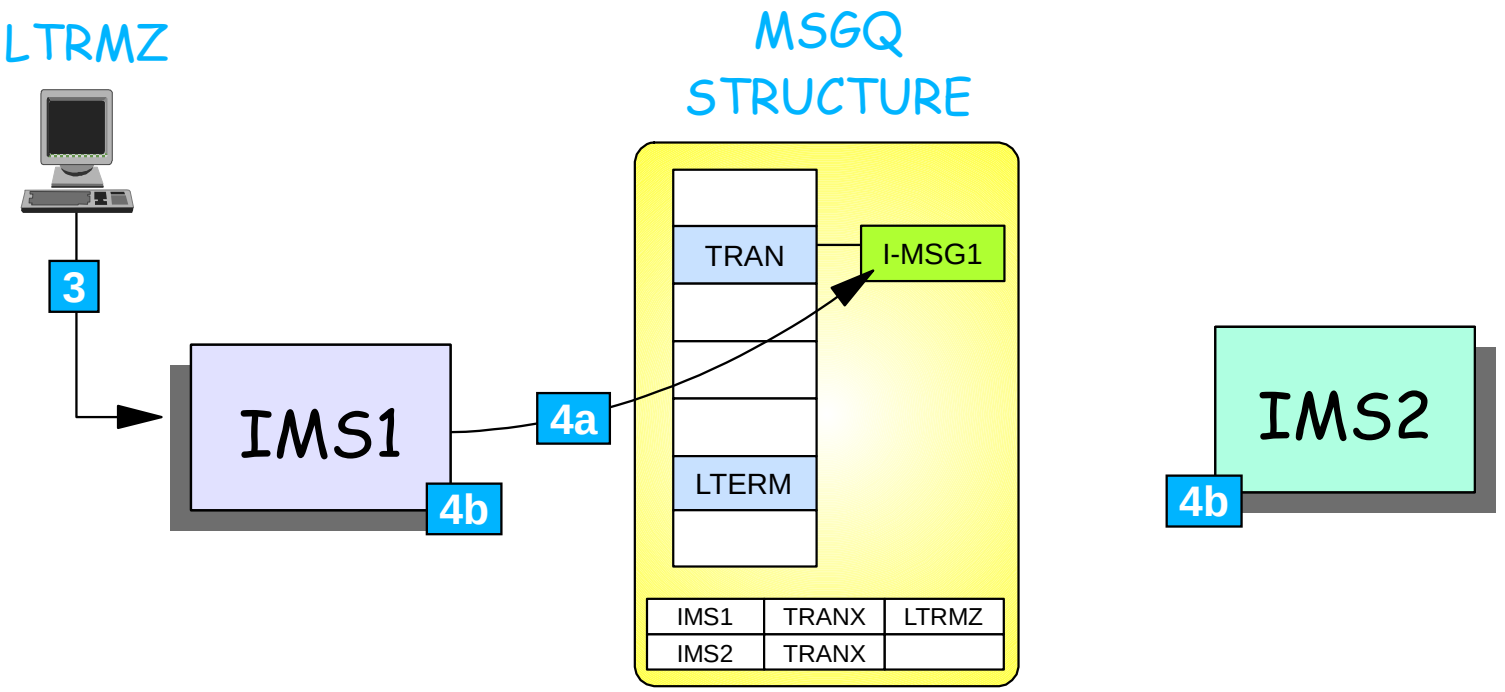
1. At initialization, IMS1 and IMS2 connect to MSGQ structure and register interest in TRANX

2. a) LTRMZ logs on to IMS1
   b) IMS1 registers interest in LTRMZ

IMS1 and IMS2

APPLCTN	PSB=PSBX
TRANSACT	CODE=TRANX
TERMINAL	NAME=NODE1
NAME	LTRMZ

Shared Message Queue Example ...

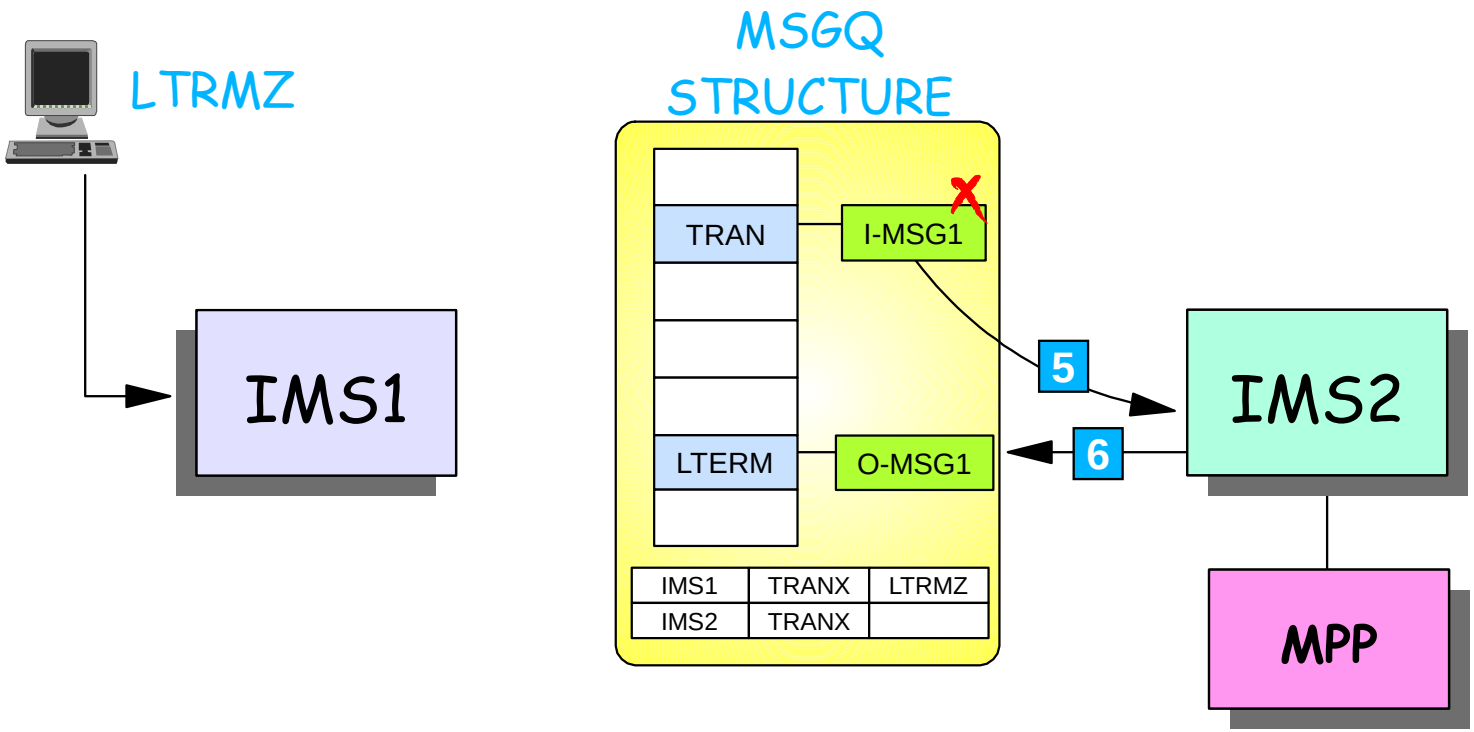


3. LTERMZ sends I-MSG1 (TRANX) to IMS1

4. a) IMS1 places I-MSG1 on TRANX queue

   b) IMS1 and IMS2 are notified (through CQS) that there is work on the TRANX queue.

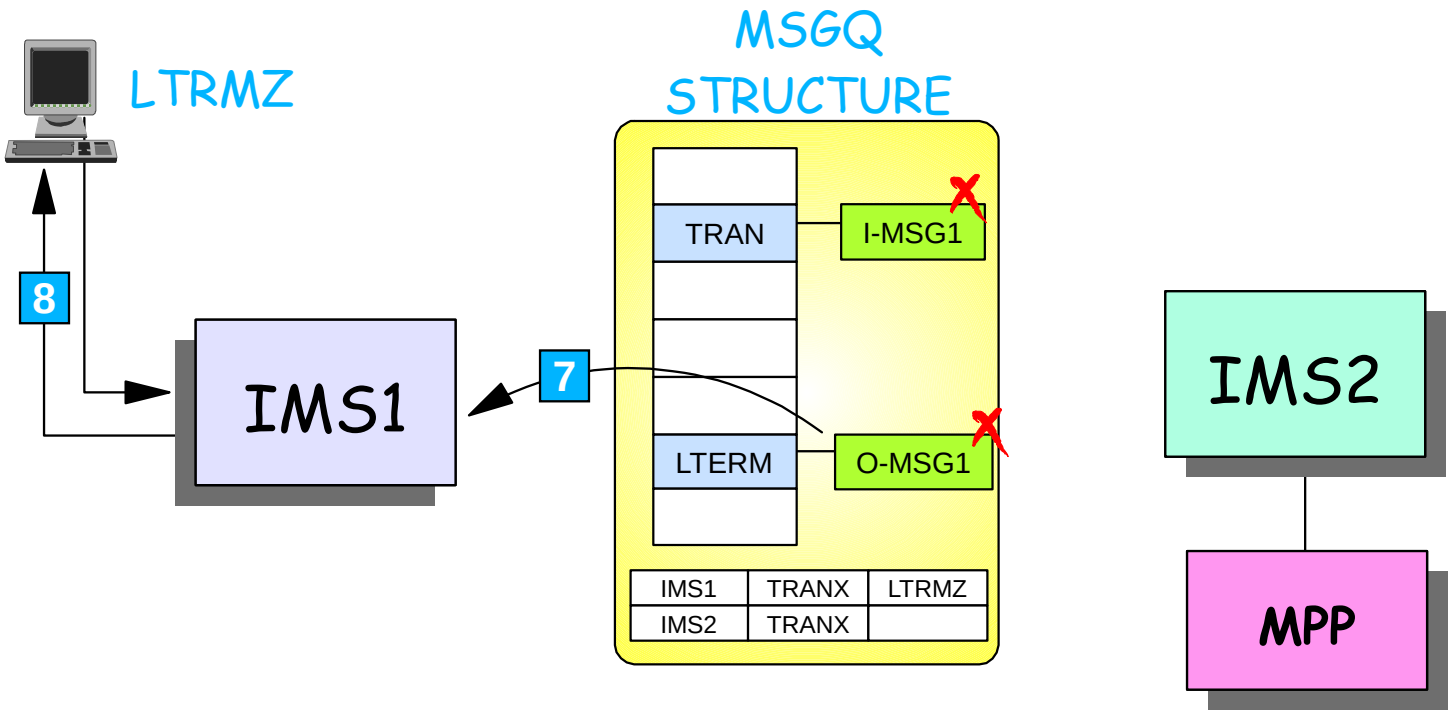
Shared Message Queue Example ...



5. IMS2 has an available MPP and retrieves I-MSG1 from TRANX queue

6. MPP processes I-MSG1 and IMS2 puts response (O-MSG1) on LTRMZ queue; I-MSG1 is deleted

Shared Message Queue Example ...

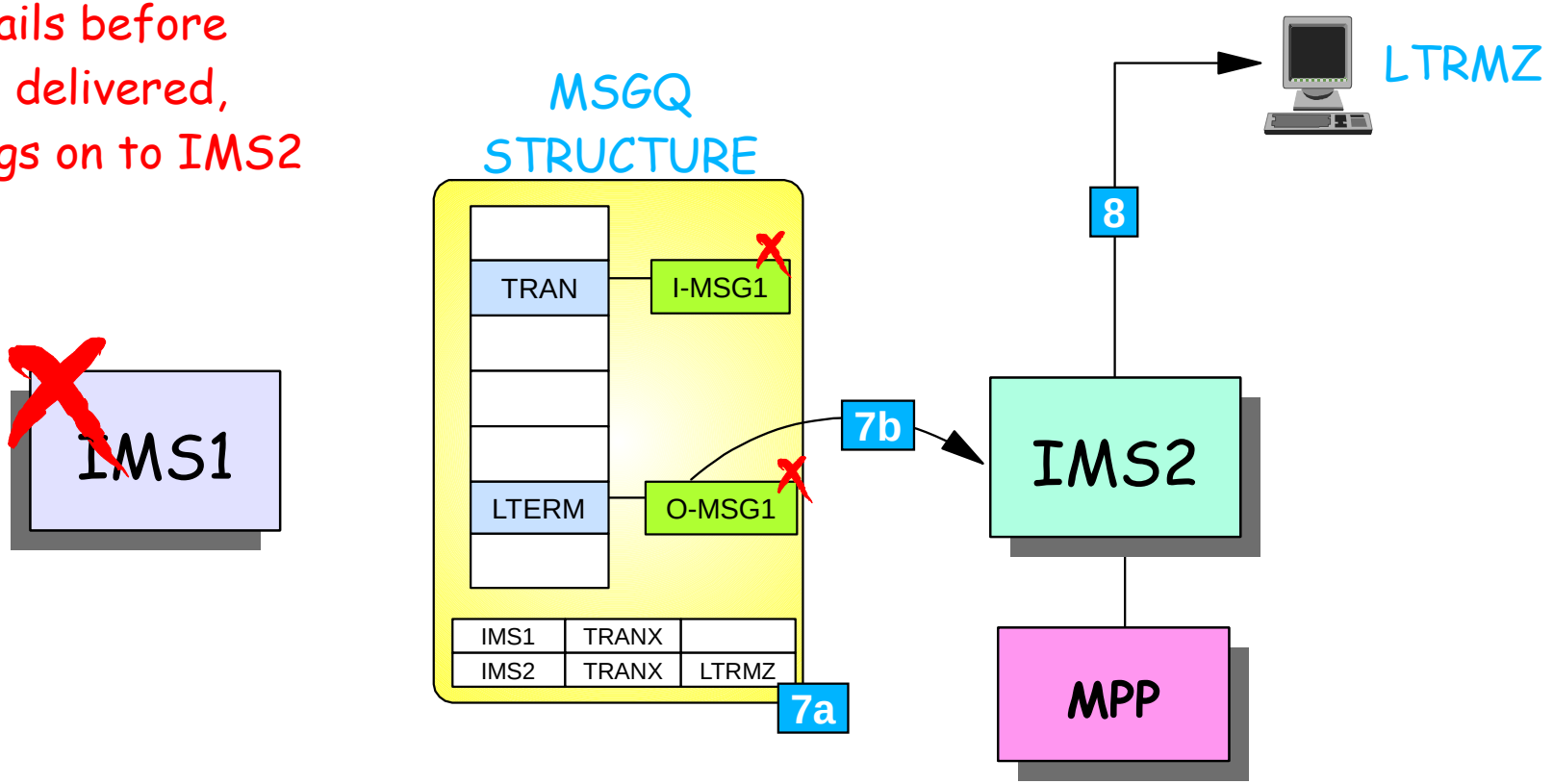


7. IMS1 retrieves response (O-MSG1) from LTRMZ queue

8. IMS1 sends response to LTRMZ; O-MSG1 is deleted when LTRMZ acknowledges.

Shared Message Queue Example ...

If IMS1 fails before response is delivered, and LTRMZ logs on to IMS2



7. a) IMS2 registers interest in LTRMZ
   b) IMS2 retrieves response (O-MSG1) from LTRMZ queue

8. IMS2 sends response to LTRMZ;
   O-MSG1 is deleted when LTRMZ acknowledges.

What's New?

All message types are available for IMSplex-wide processing

- IMS V6
  - No support for APPC and OTMA transactions
  - Had to execute on front-end IMS
- IMS V7
  - Added support for asynchronous APPC and OTMA transactions
- IMS V8
  - Adds support for synchronous APPC and OTMA transactions

What's New in Version 8?

When running IMS V8 with ...

- Common Service Layer
- Resource Management Structure
- Shared Queues

We get ...

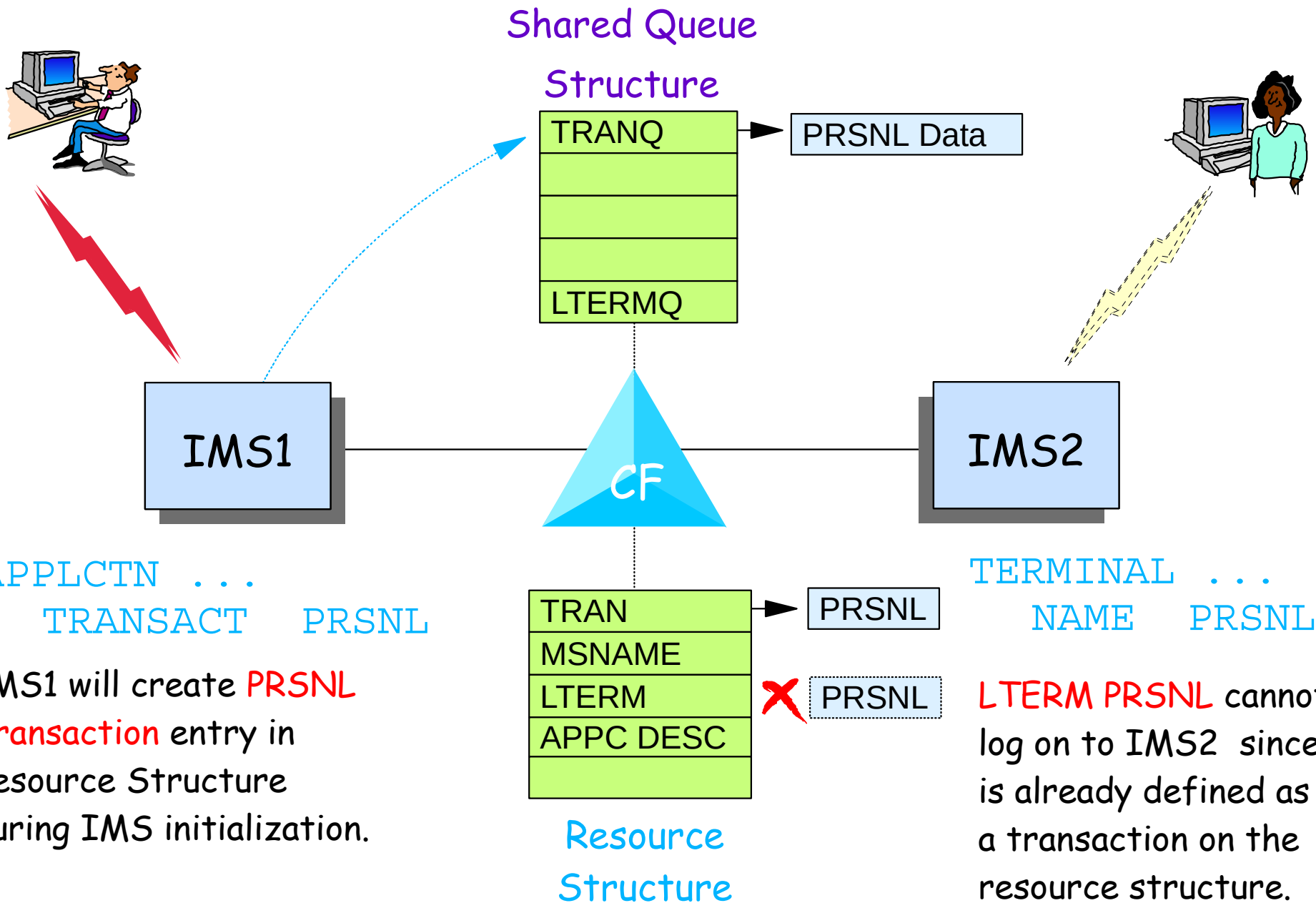
- Sysplex Terminal Management
  - Resource type consistency
    - Can't have different resource types with same name
  - Resource name uniqueness
    - Resource can be active on only one IMS at a time
  - Resource status recovery
    - If session terminates with significant status (e.g., in a conversation), can resume that status (the conversation) on any other IMS in the IMSplex

Sysplex Terminal Management

Sysplex terminal management objectives

- Enforce global *resource type consistency*
  - Prevent naming inconsistencies between IMSs
- Enforce global *resource name uniqueness*
  - Prevent multiple logon / signon within the IMSplex
- Enable global *terminal and user resource status recovery*
  - Resume significant status on another IMS after failure
    - Conversation, fast path response, STSN sequence numbers
    - Command status (e.g., stopped, assigned, ...)
  - Reduce need for IMS-managed VGR affinity
- Enable *global callable services*
  - User exits can access terminal and user information across IMSplex

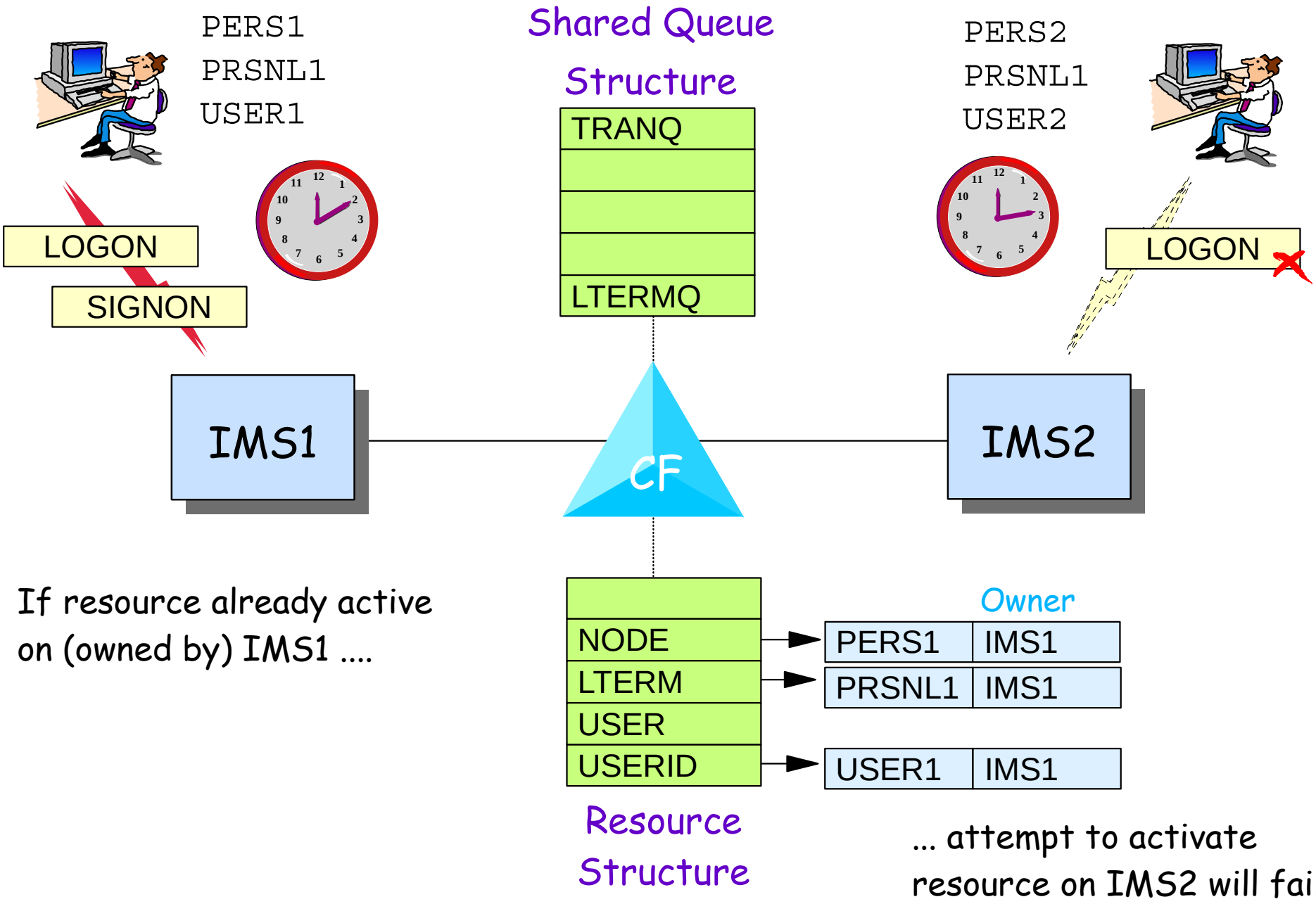
Resource Type Consistency



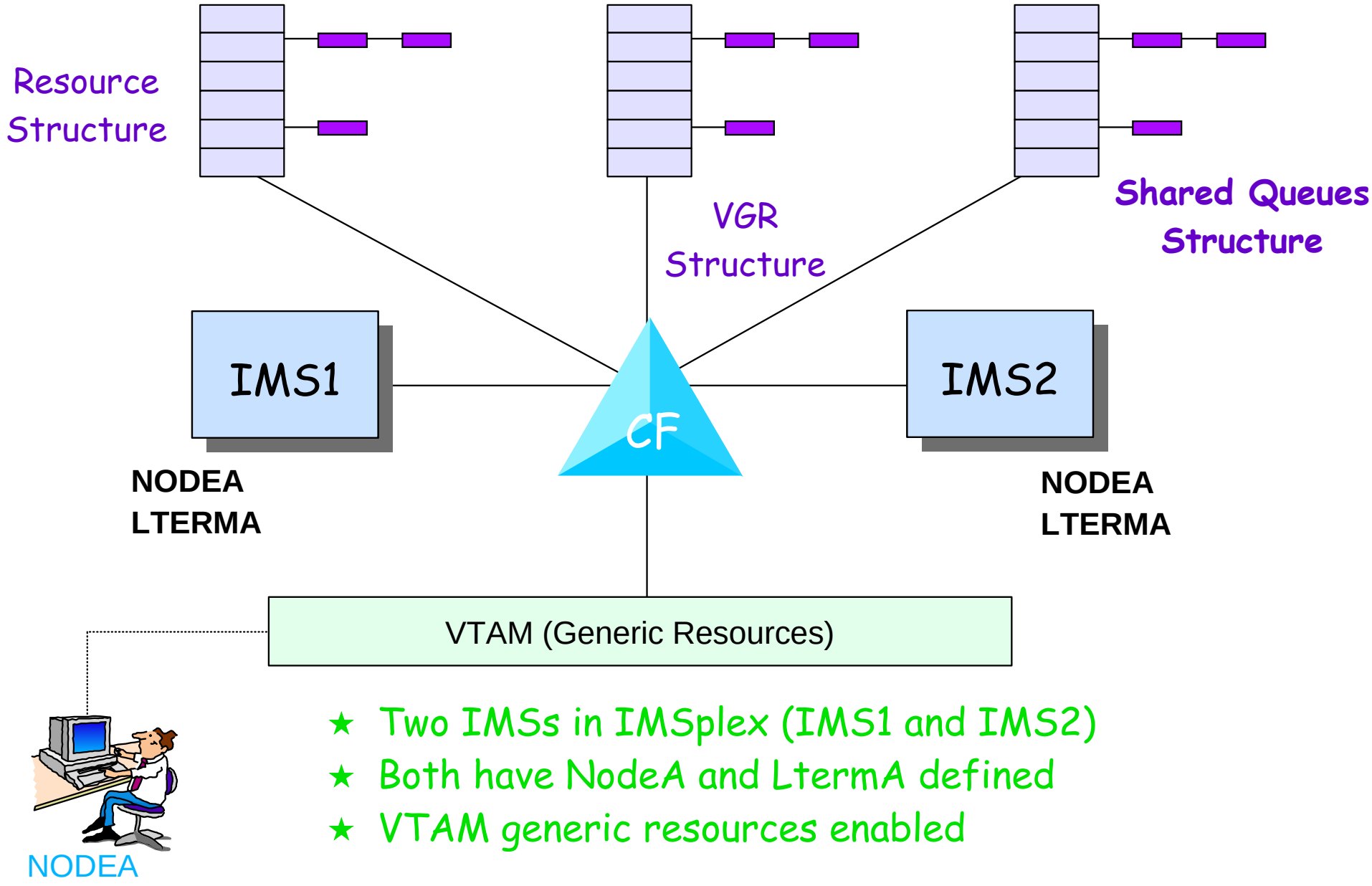
IMS1 will create PRSNL Transaction entry in Resource Structure during IMS initialization.

LTERM PRSNL cannot log on to IMS2 since it is already defined as a transaction on the resource structure.

Resource Name Uniqueness

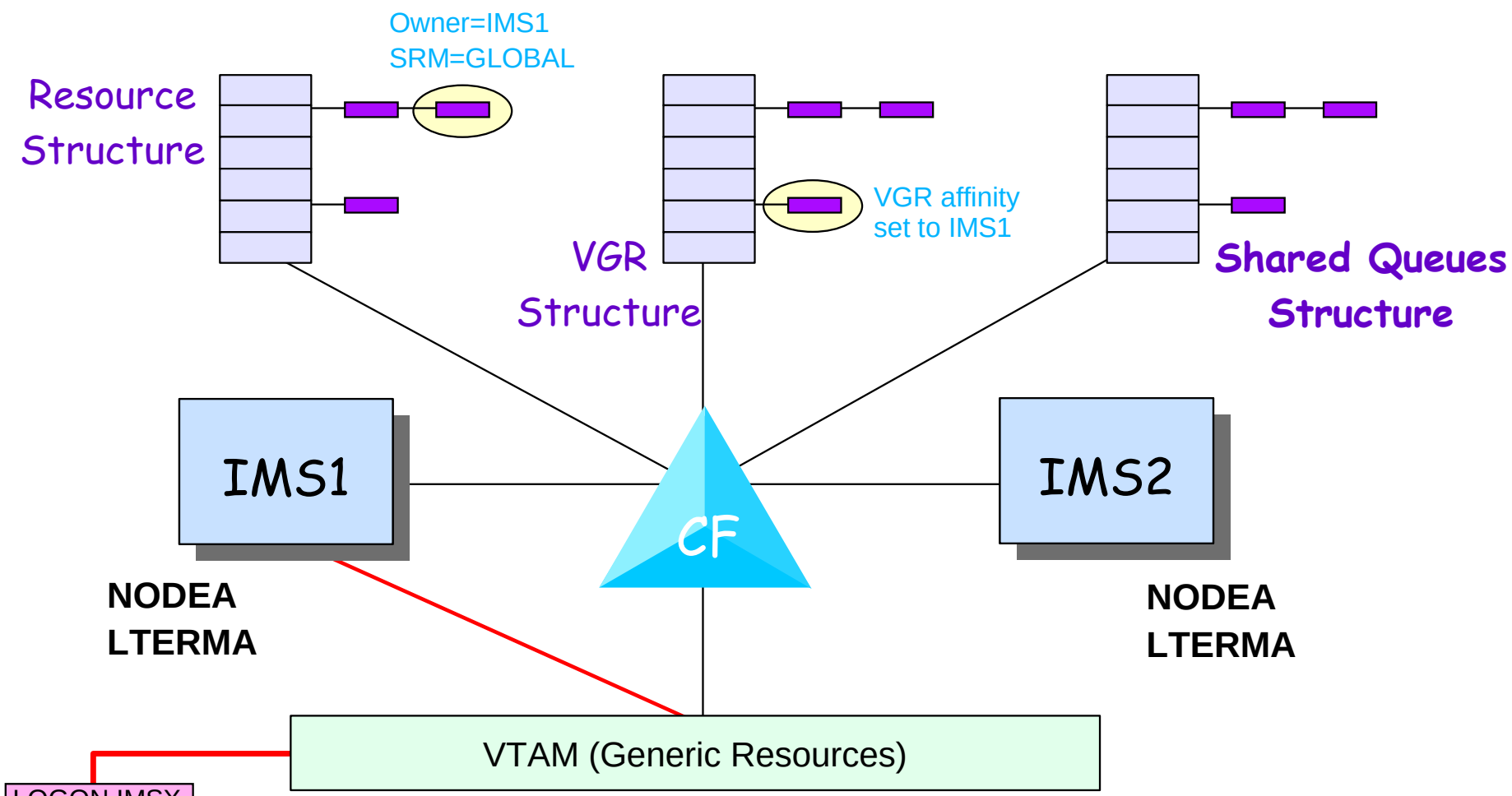


Status Recovery Example



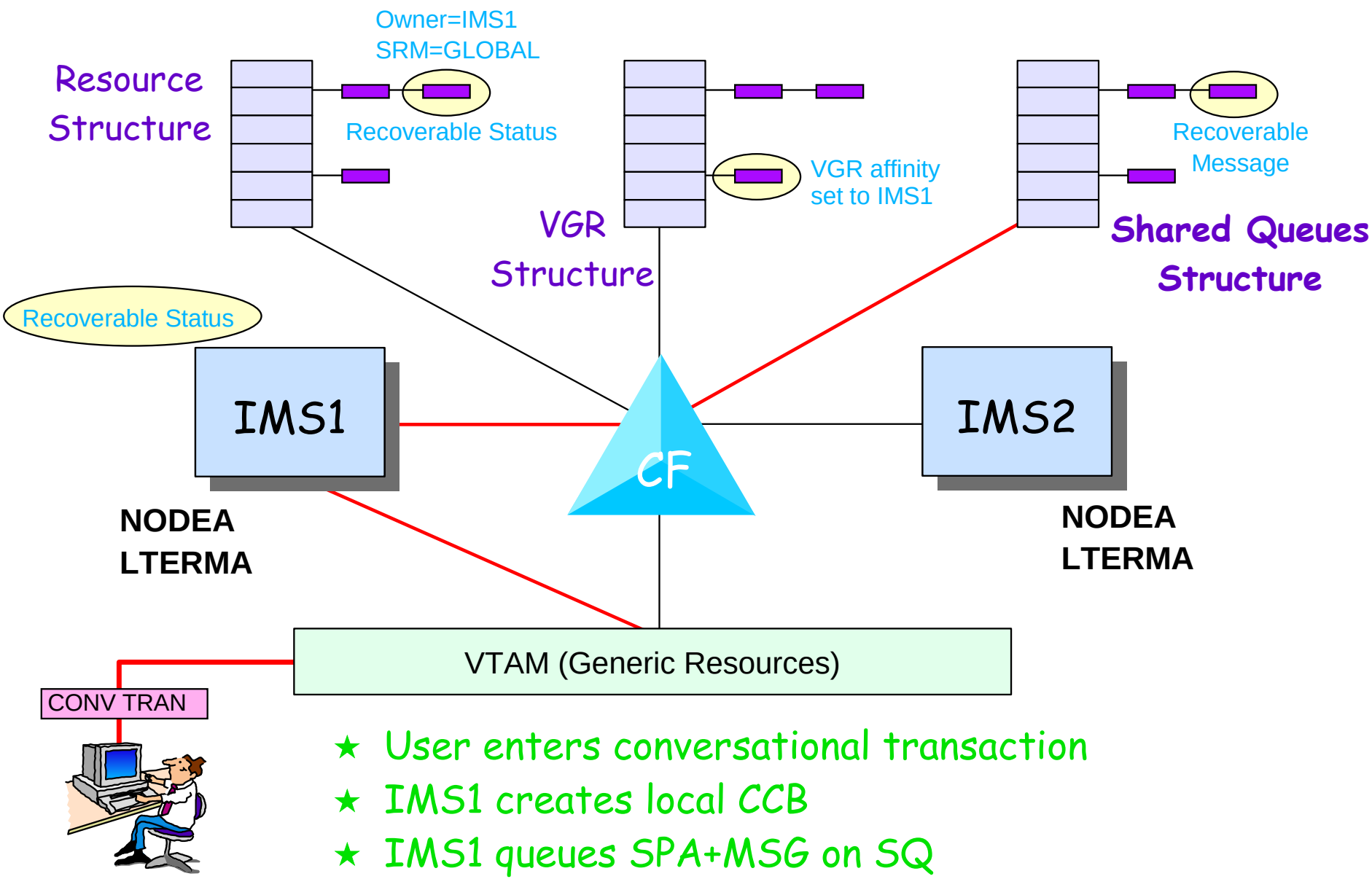
- ★ Two IMSs in IMSplex (IMS1 and IMS2)
- ★ Both have NodeA and LtermA defined
- ★ VTAM generic resources enabled

Global Status Recovery



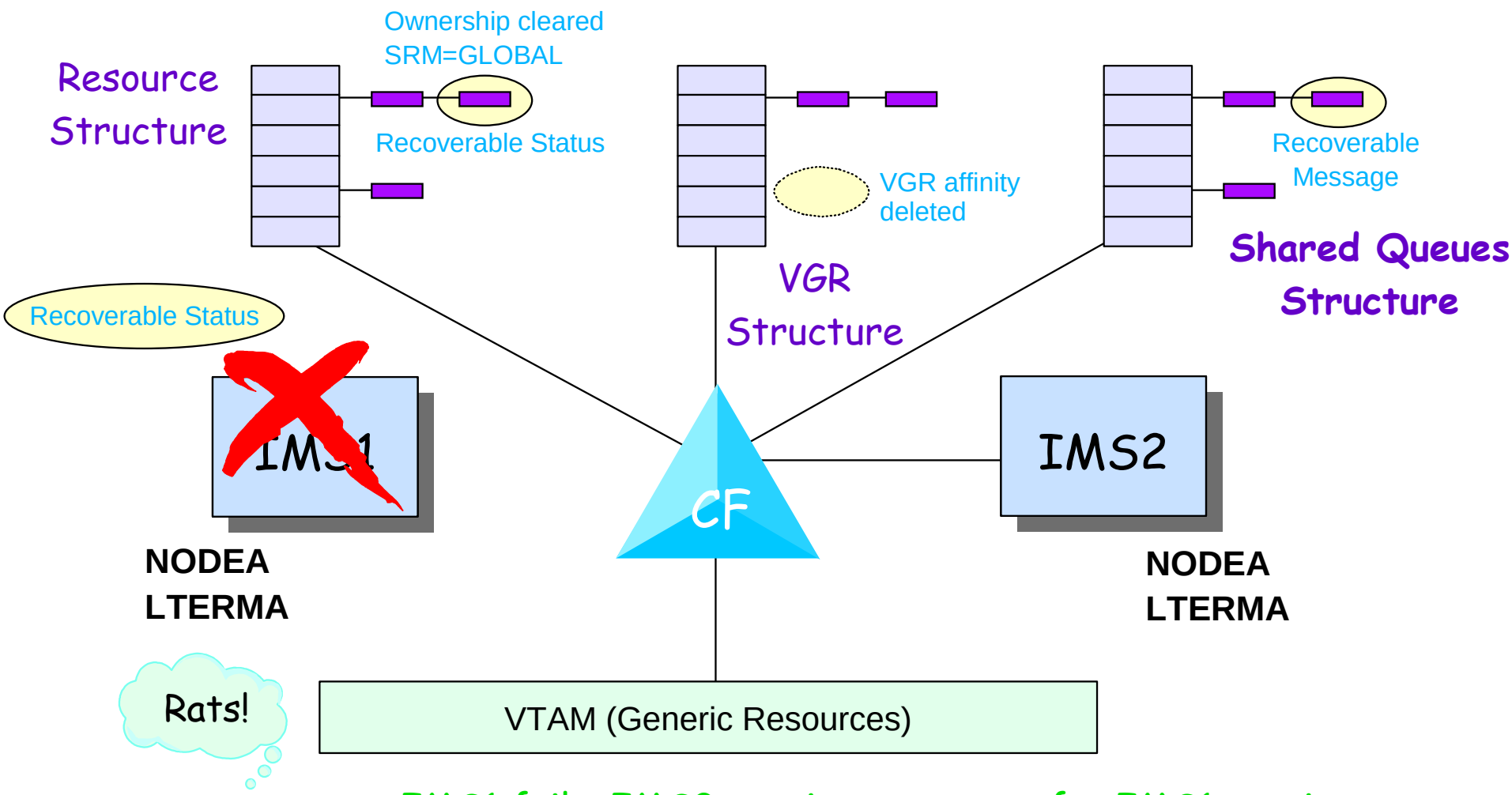
- ★ End-user logs on using generic logon IMSX
- ★ VTAM routes logon to IMS1; sets VGR affinity to IMS1
- ★ IMS1 sets SRM=GLOBAL
- ★ IMS1 sets VGR affinity management to VTAM
- ★ Resource entries created: Owner=IMS1, SRM=GLOBAL

Global Status Recovery ...



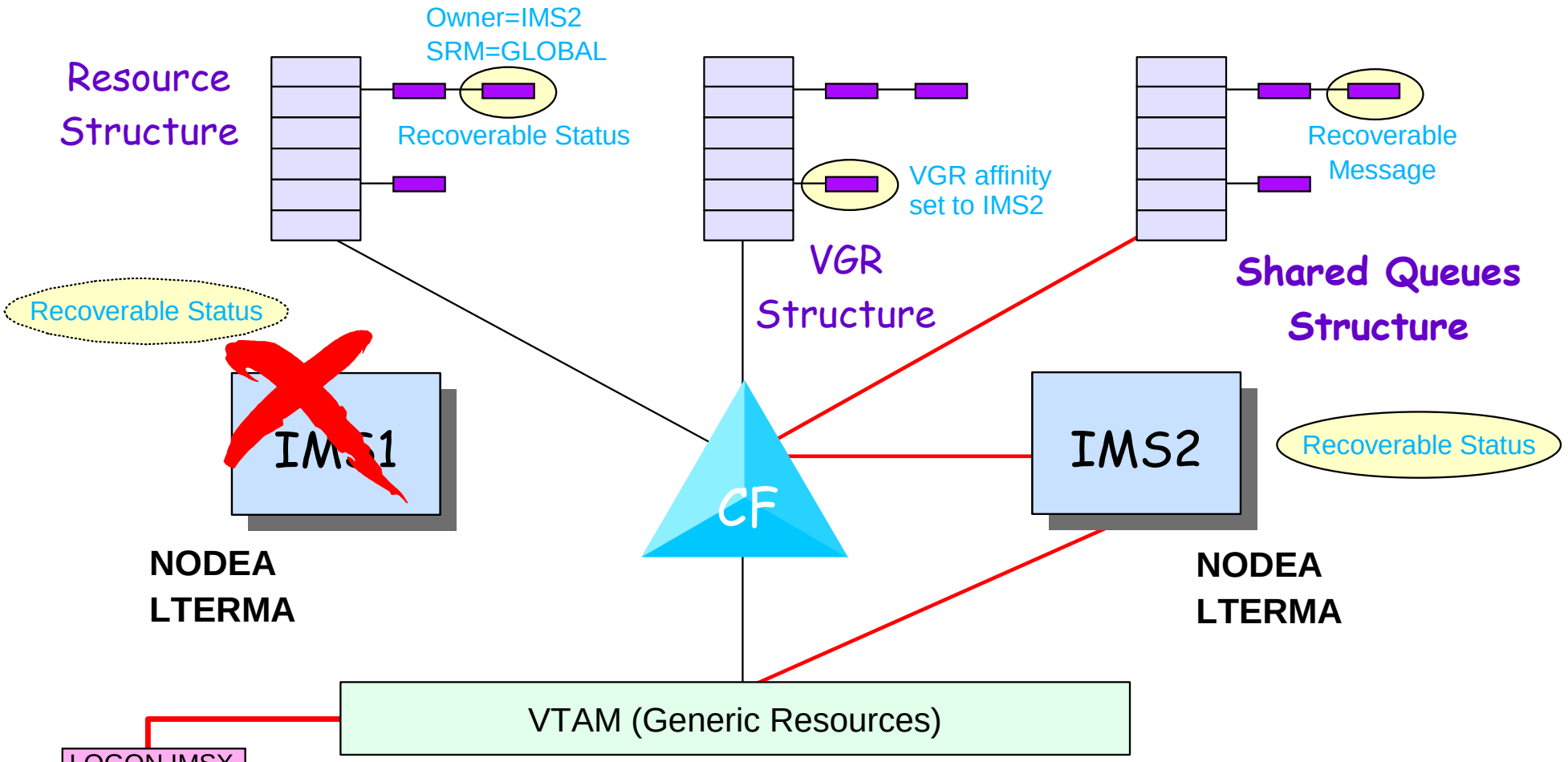
- ★ User enters conversational transaction
- ★ IMS1 creates local CCB
- ★ IMS1 queues SPA+MSG on SQ
- ★ IMS1 updates user resource with Conv-ID

Global Status Recovery ...



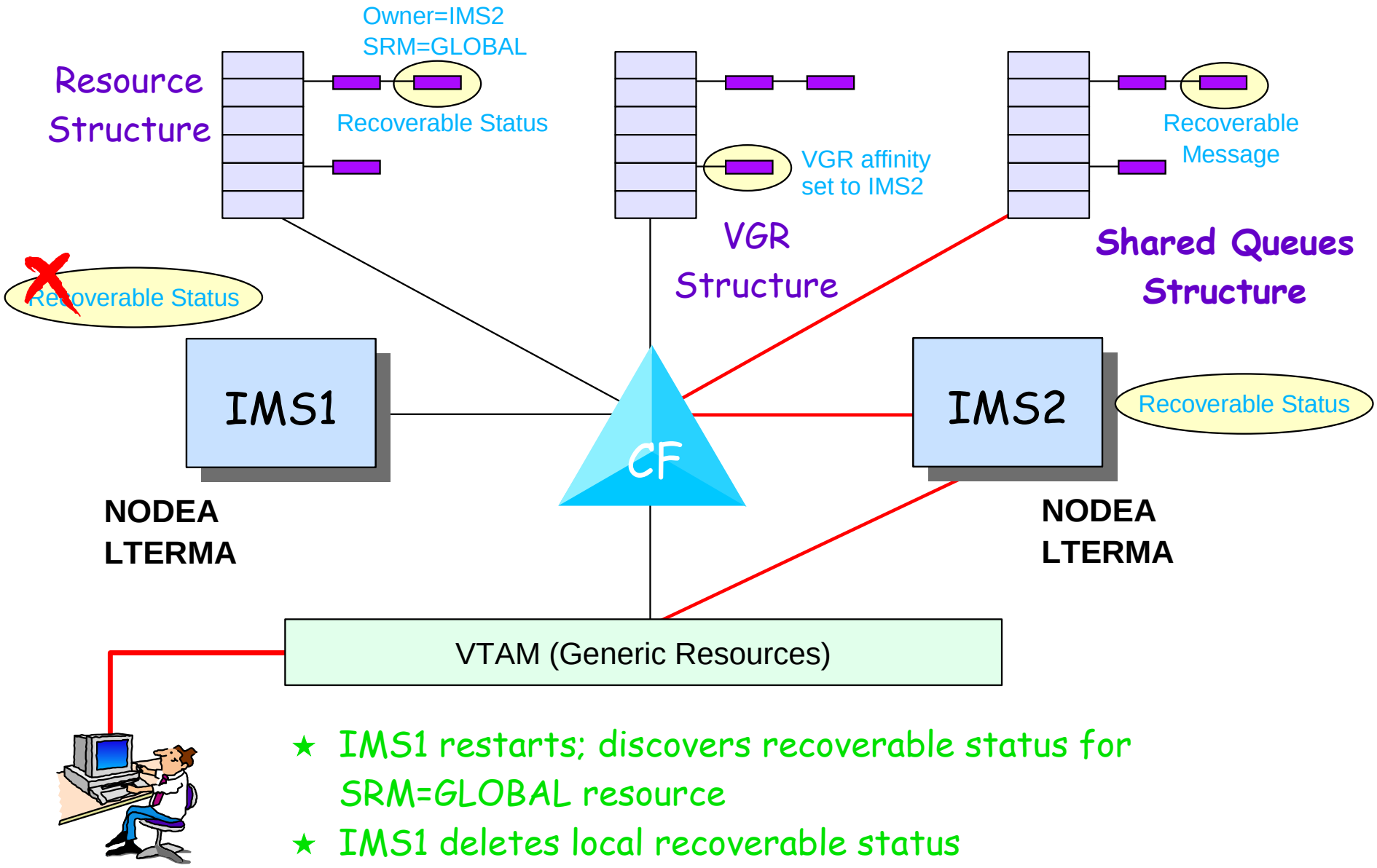
- ★ IMS1 fails; IMS2 queries structure for IMS1 entries
- ★ IMS2 does not delete resource entry (in conversation)
- ★ IMS2 clears ownership (SRM=GLOBAL)
- ★ VTAM deletes VGR affinity
- ★ SPA+MSG still on SQ

Status Recovery - SRM=GLOBAL ...



- ★ End-user logs on using generic logon IMSX
- ★ VTAM routes logon to IMS2; sets VGR affinity to IMS2
- ★ IMS2 sets ownership to IMS2
- ★ IMS2 recovers conversation from Resource Structure and Shared Queue Structure

Status Recovery - SRM=GLOBAL ...



- ★ IMS1 restarts; discovers recoverable status for SRM=GLOBAL resource
- ★ IMS1 deletes local recoverable status

Cloned IMSs

Cloned IMSs can use same system definition

- Each IMS is capable of processing any transaction (or BMP)

Shared Queues does not require cloned IMSs, but …

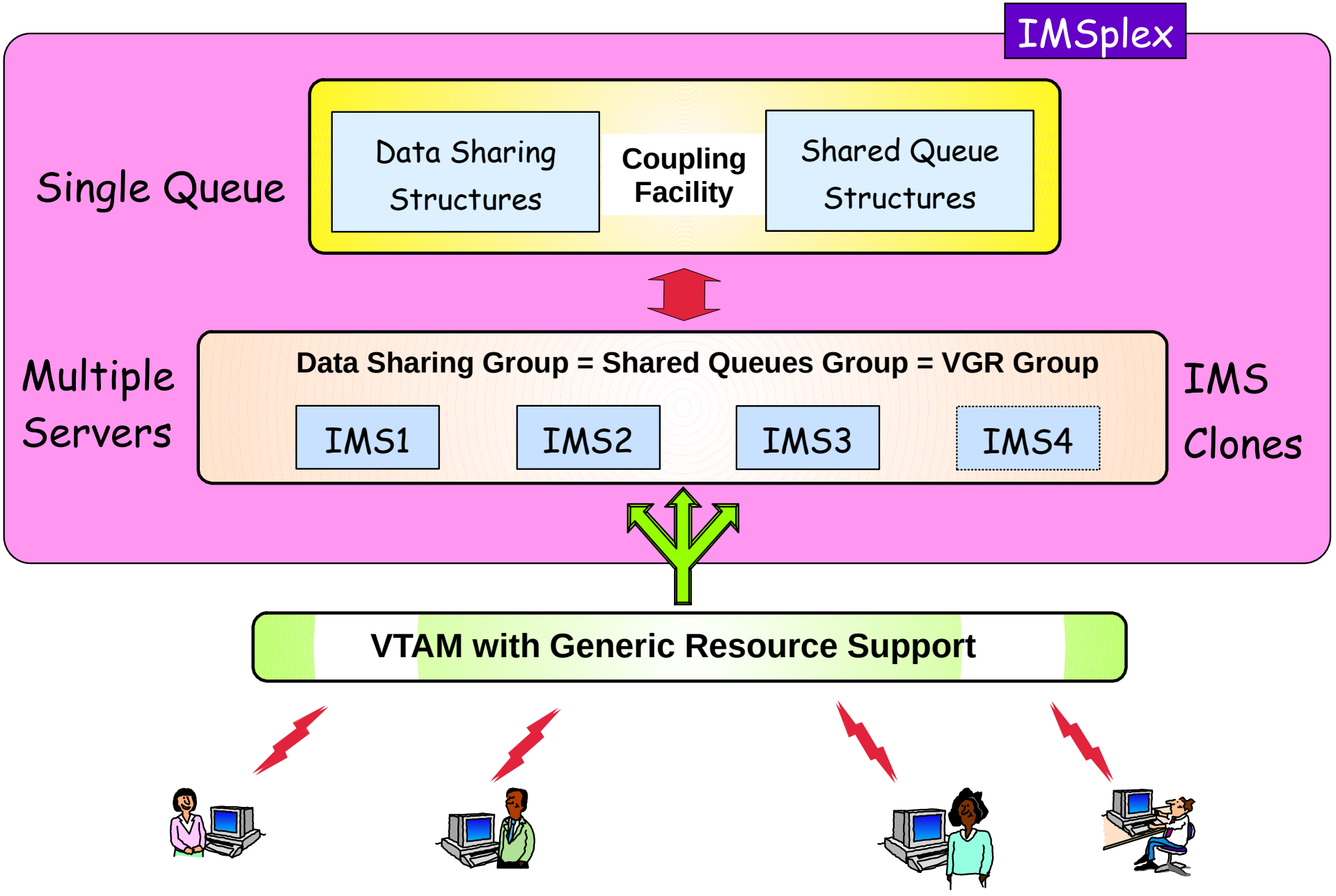
- IMS will not queue input message if transaction not defined
  - User is sensitive to which IMS he/she logged on to
- Diminishes load balancing
  - Not all IMSs can process all transactions
- Diminishes availability
  - If the only IMS with trancode defined is down
- Diminishes capacity
  - Long queues cannot be processed by other systems

Cloned IMSs ...

If used in combination with VTAM Generic Resources

- User logs on to generic name
- VGR routes logon request to IMS with fewest logons
  - Network load balancing
- User enters transaction
  - IMS puts transaction on shared queue
- If that IMS fails before transaction processed
  - User can logon again using generic name
    - No need to wait for IMS emergency restart
  - VTAM routes logon request to available IMS
  - New IMS can process transaction and send response
    - Including response for transaction entered from failed IMS

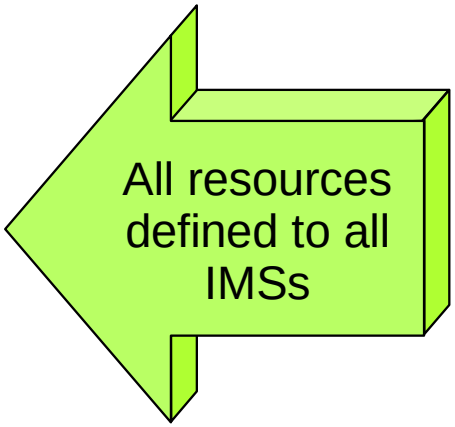
Cloned IMS Configuration



Cloned IMS Configuration ...

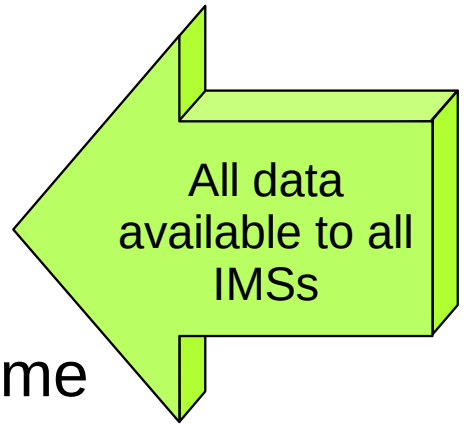
IMS System Definition

- Databases
- PSBs/Transactions
- Network
  - Terminals and/or ETO
  - APPC and OTMA definitions
- MSC links
  - Define all links to all IMSs



Define data sharing environment

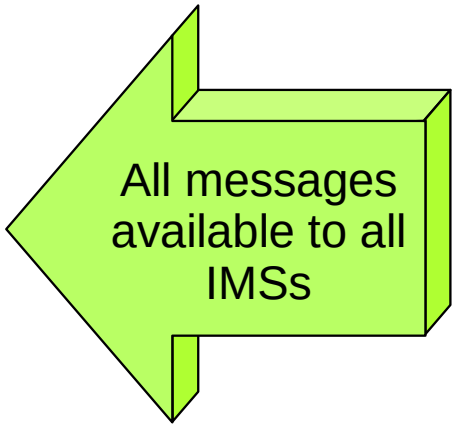
- Register databases at share level 3
- IMSGROUP parameter (for BMP connectivity)
- Define IRLMs with same data sharing group name
- Define data sharing structures to IMS and IRLM
- Define FDBR environment



Cloned IMS Configuration ...

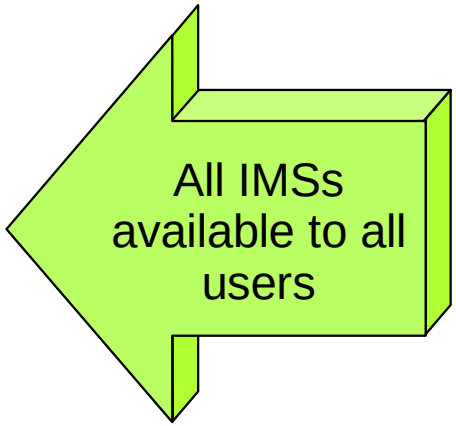
Define shared queues environment

- IMS and CQS proclib members
  - CQS address spaces
  - SQ Structures
  - Logstream
- Shared queue XCF groups
  - IMS - SQGROUP
  - CQS - CQSGROUP



VTAM Generic Resources

- GRSNAME for each IMS (non-APPC)
- APPC generic resources supported by APPC/MVS



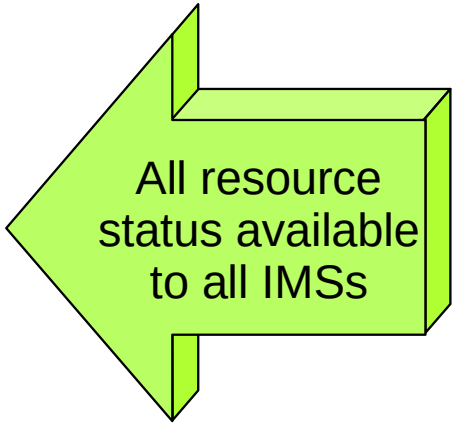
Sysplex distributor

- For TCP/IP connections

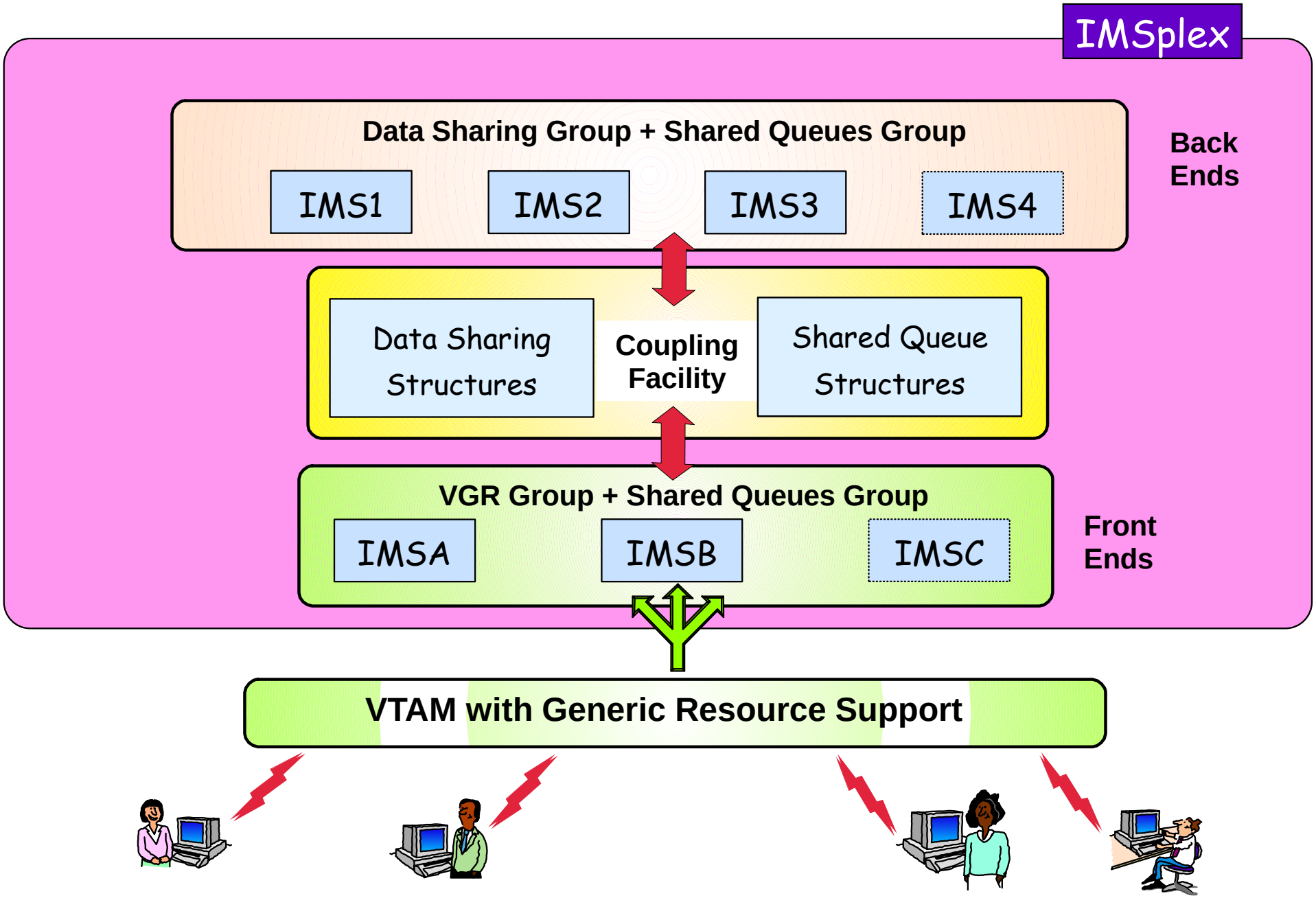
Cloned IMS Configuration ...

Define Common Service Layer (IMS V8)

- Structured call interface
  - Intra-IMSplex communications
- Operations manager
  - Single point of control
- Resource manager with RM Structure
  - Sysplex terminal management
  - Coordination global online change
  - Global callable services



Front-End / Back-End Congfiguration



FE/BE Configuration ...

Advantages

- Back-end processing IMSs have no network connections
- Back-ends can be added, removed without impacting end-user
- Back-end failure impacts only transactions currently scheduled
  - Queued transaction process on another IMS
  - End-users not connected to back-end

Disadvantages

- All transaction processing done in back-end
- No "front-end processing" advantage
  - Full function "local"
  - Fast path "local first"

Benefits of Shared Queues

Automatic work load balancing

- A message placed on the Shared Queues can be processed by any IMS with interest in the message
- Only IMS with current capacity will request (pull down) work

Incremental growth / capacity

- New IMS subsystems can be added as workload increases
- New IMSs can be for processing only (no network) during periods of heavy activity (all configurations)

Improved availability

- If an IMS fails, the workload may be assumed by the surviving IMSs
- End user can continue on surviving IMS
- Shared queues are not lost if one or more IMSs are cold started